11-4-2011

# Distributed reactive routing for selective forwarding attack resilience in wireless sensor networks

Jonathan Szymaniak

# Distributed Reactive Routing for Selective Forwarding Attack Resilience in Wireless Sensor Networks

by

**Jonathan T. Szymaniak**

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science
in Computer Engineering

Supervised by

Associate Professor & Department Head Dr. Shanchieh Jay Yang
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
November 4 2011

Approved by:

---

Dr. Shanchieh Jay Yang, Associate Professor & Department Head
*Thesis Advisor, Department of Computer Engineering*

---

Dr. Andrés Kwasinski, Assistant Professor
*Committee Member, Department of Computer Engineering*

---

Dr. Marcin Łukowiak, Assistant Professor
*Committee Member, Department of Computer Engineering*

# Dedication

To Brian, Becky, Jessica, Jason, and Brittany -
Words cannot sufficiently express my gratitude for your love and support.

# Acknowledgments

I would like to sincerely thank my advisor, Dr. Shanchieh Jay Yang, for his guidance, support, and mentoring throughout my time at RIT. His inspiring encouragement and enthusiasm made it possible for me to complete this work.

I would also like to thank my committee members, Dr. Andrés Kwasinski and Dr. Marcin Łukowiak, for reviewing my work and providing invaluable advice.

Lastly, I would like to acknowledge the maintainers of Castalia, Athanassios Boulis & Yuri Tselishchev (NICTA), and the authors of CTP-Castalia, Ugo Colesanti (Sapienza Università di Roma) & Silvia Santini (ETH Zurich), for their quick responsiveness to questions and for supporting bugfixes.

# Abstract

**Distributed Reactive Routing for Selective Forwarding Attack Resilience in Wireless Sensor Networks**

**Jonathan T. Szymaniak**

**Supervising Professor: Dr. Shanchieh Jay Yang**

Wireless Sensor Networks (WSNs) are an emerging technology that may one day supply real-time information to many services and Internet-based applications. The utility of WSNs relies on the ability to provide valid information, even in the presence of failures or attackers. Current research in the field has identified a large variety of attacks and countermeasures, however, few works address how WSN routing protocols can autonomously react to detected attacks. The works that do provide attack-reactive schemes generally require nodes to coordinate or exchange trust/detection reports.

This work aims to maximize data delivery in the presence of selective forwarding attacks with nodes performing detection and reaction operations independently. Via modifications to the Collection Tree Protocol's forwarding path and route update procedure, nodes autonomously evaluate their parent's forwarding reliability and duplicate data to alternative parent nodes when deemed necessary. As shown though a number of simulations, this distributed scheme yields significant data recovery with only modest overheads for attackers dropping data at medium to high rates.

# Contents

# List of Tables

# List of Figures

# Glossary

## B

**Base Station** In WSN literature, this term often refers to a location containing both a sink node in the WSN and the data processing infrastructure (e.g., a desktop computer).

## C

**CTP** Collection Tree Protocol. A tree routing protocol designed for TinyOS. Notable aspects of this protocol include high data delivery rates, a link quality-based routing gradient, adaptive route beaconing, addressless routing, mulit-gateway support, and routing loop detection and resolution.

## G

**Gateway** In most WSN literature, this term is synonymous with "Base Station." However, this work uses the term *Gateway* to refer to a sink node in the network that transmits data to another location for further processing via an out-of-band communications channel. This slight difference in nomenclature addresses the Sensor-Cloud paradigm in which multiple Gateways in a network may contribute received data to one or more Cloud systems.

## M

**Mote** Sensor node platforms are typically referred to as "motes." For examples of such platforms, see:
`http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html`.

## P

**PRE** Parent Reliability Estimator. The mechanism introduced in this work that allows nodes to monitor the data forwarding reliability of their parents.

**PRV** Parent Reliability Value. The metric associated with the PRE that quantifies a parent's forwarding reliability. This metric is represented by the symbol, $\rho$, and has a nominal value of 1.0; increasing values indicate a less reliable forwarder.

**S**

**Selective Forwarding Attack** An attacker correctly forwards some, but not all data messages. This may have the effect of biasing the perception of the monitor environment in the attacker's favor.

**T**

**THL** Time Has Lived. A field in CTP data messages that denotes how many hops a message has traversed.

**W**

**WSN** Wireless Sensor Network. A network of low-power embedded devices, equipped with one or more sensors, monitoring an environment an propagating sensed information toward one or more sinks in the network for further processing. Communication is typically multi-hop, and 802.15.4 transceivers are commonly used.

# Chapter 1

# Introduction

Wireless Sensor Networks (WSNs) consist of low-power and low-cost embedded platforms typically equipped with a microcontroller, a transceiver, and one or more sensors. These devices are used to gather sensor data pertaining to an activity or phenomena of interest and propagate this information toward one or more base stations for further processing. Due to the relatively short transmission distance capabilities of sensor node transceivers, messages are often relayed via a number of intermediate nodes.

The low cost of embedded sensor platforms and the highly distributed nature of communication in ad-hoc wireless networks allow WSNs to monitor large geographical areas or harsh environments that are difficult to cover with traditional wired network infrastructures. For example, WSNs have been used to monitor the movement of glaciers, species population trends, and river levels for early flood detection [1]. Other applications of WSNs include surveillance systems, medical patient monitoring, home and office automation, theft detection, and industrial process monitoring [2][3].

With the ever decreasing cost of electronics, WSNs have great potential to become a pervasive technology. As the Cloud Computing[1] paradigm has become popular in recent years, a number of researchers have begun investigating how Cloud Computing may be used to access, store, process, and distribute the vast amount of information obtained by WSNs [5][6]. In envisioning the future of data collection, processing, and dissemination to end users, it is clear that the principal asset in Sensor-Cloud systems is valid information. If end users are expected to make decisions utilizing information gathered by WSNs, services must ensure the correctness of this information, especially in the presence of failures or malicious users attempting to influence perceptions of

[1] A detailed definition of "Cloud Computing" can be found in [4].

monitored events.

As detailed in [7], [8], [9], and [10], there are numerous methods by which attackers can interfere with the operations of a WSN. To gain access to a network, an attacker may physically compromise a node (which are often assumed to be unattended) and reverse engineer the device's firmware to extract network secrets (e.g., cryptographic keys) and identify software vulnerabilities. Upon installing malicious software on a network node, an attacker may then perform eavesdropping and traffic analysis attacks to identify ideal locations in the network to mount an attack. Once an attacker has one or more nodes participating in the network, he or she may then perform a variety of attacks (e.g., rushing, sink hole, worm hole, Sybil) to acquire significant influence over the flow of data in the network. In many cases, this influence is associated with becoming a highly utilized member of a routing path. Once an attacker controls the flow of information, he or she may tamper with or disrupt the flow of data. This work will focus on a particular type of data flow disruption, referred to as a "selective forwarding attack." In this attack, a malicious node will forward some data correctly, while dropping other messages. This type of attack may allow an attacker to suppress the delivery of information that he or she does not want to be made accessible to end users or services. If an attacker can suppress a significant amount of data, he or she may be able to bias perceptions of the monitored events in his or her favor.

Defenses against the following attacks generally fall into the categories of prevention, detection, and resilience through multipath routing. Prevention schemes tend to include either cryptographic authentication schemes that prevent unauthorized nodes from participating on a network [11] [12] [13] [14], or countermeasures for specific attacks [10] [15] [16]. Attack detection schemes seek to detect the presence of malicious nodes by identifying anomalies [17] [18], or specification/rule violations [19] [20]. Multipath routing provides inherent resilience to failures and data flow attacks by duplicating data along multiple braided or disjoint paths, with the inevitable trade-offs of increased network traffic and increased power consumption [21].

Although a great deal of work has been done in these areas, few works have addressed how detection mechanisms may be used to facilitate distributed reactive routing decisions, while targeting high data delivery in the presence of attackers. The work of Sun *et al.* [22] details an attack-reactive routing scheme based a trust management framework that involves nodes establishing, evaluating, and reporting the trustworthiness of surrounding nodes. Using these trust metrics, Sun *et al.* showed, using

hardware-based experiments, that such a framework can be effectively used to exclude untrustworthy nodes from routing paths. However, as described in [22], the problem of establishing and maintaining trust between nodes is a difficult problem, especially in cases where attackers target the trust mechanism itself. [23] presents a scheme in which nodes respond to detected attacks by initiating a clustering algorithm that excludes the malicious node from the cluster, and attempting to wirelessly reprogram the malicious node. However, by utilizing schemes that introduce additional interactions between nodes, the overall system's attack surface is increased, possibly yielding new opportunities for an attacker. The primary goal of this work is to determine if an effective attack-reactive scheme for selective forwarding attack resilience can be achieved through autonomous and distributed decision-making, rather than a coordinated effort between multiple nodes. This work also investigates the effectiveness of opportunistically choosing secondary forwarder nodes only on an as-needed basis, as opposed to explicit multipath construction as done in other works.

# Chapter 2

# Background And Supporting Work

This chapter presents background information pertaining to existing work in WSN security and the routing protocol used as a starting point for this work. First, an overview of WSN threats is presented, followed by a discussion of associated countermeasures provided by current work in the field. Next, the important features of the Collection Tree Protocol are briefly described, and the opportunity for mounting a selective forwarding attack on this protocol is presented.

## 2.1 Threats Facing Sensor Networks

As reported in [7] [8] [9], and [10], numerous classes of attacks on WSNs have been explored. A summary of these threats is presented in Table 2.1, grouped by the stages of an attack in which they might be utilized. Note that these categorizations of attacks are not intended to describe a definitive attack model; they are presented here only to conceptualize a possible procedure by which a WSN attack may be mounted.

The attacks labeled *Reconnaissance & Preparation* allow an attacker to first passively study the network and then begin interacting with a subset of nodes, in order to acquire the ability to actively participate on the network. As nodes are often assumed to be left unattended, physical tampering or removal of a small number of nodes from the network is considered a realistic and plausible threat. Eavesdropping may allow an attacker to determine the optimal deployment of attacking nodes in order to achieve his or her desired goals.

Upon gaining the ability to participate on the network, an attacker may need to perform a preliminary set of actions to ensure the desired impact on the network is feasible and that various protocol mechanisms can be effectively circumvented. Some

of these actions are presented in the *Acquiring Influence* section of Table 2.1. In the case of this work, "influence" is generally associated with a malicious node becoming a trusted member of a heavily used routing path, providing an attacker with information to steal, manipulate, or suppress. To become a member of one or more routing paths, an attacker may perform a Sybil attack to fill neighbor nodes' routing tables with aliases for a malicious node, or to bad-mouth other viable neighbors. Alternatively, an attacker may utilize a Sinkhole attack by reporting the availability of a route that is shorter or more reliable than that of its neighbors.

Table 2.1: WSN threats, grouped by attack stage

| Attack Stage | Attack Class | Description |
|---|---|---|
| Reconnaissance & Preparation | Physical Capture | An attacker obtains physical access to a node and uploads malicious code or extracts sensitive information from the device (e.g., cryptographic keys). |
| | Eavesdropping | Attacker monitors wireless channel, attempting to overhear sensitive information and/or deduce network topology based upon traffic flow. |
| Acquiring Influence | Sinkhole | Malicious node falsely advertises high quality route in order to "pull" routes toward it. |
| | Wormhole | Attacker relays messages between malicious nodes in different parts of network using an out-of-band low latency link. This attack allows the attacker to establish a virtual route to a gateway that is significantly shorter than neighboring nodes. |
| | Sybil | Malicious node acts as, or claims to neighbor, multiple network node identities. This is particularly effective against mechanisms using a collaborative voting scheme. |
| | HELLO Flood or Rushing | Malicious node transmits route request ("HELLO") messages at a TX power greater than legitimate nodes, giving the impression of a higher quality link. If nodes do not verify the ability to perform bidirectional communication with the parent, this can result in a loss of all upstream traffic. |
| End Goal | Information Theft | Once an attacker has gained access to sensitive information, he or she may use malicious nodes to log data. |
| | Denial of Service | An attacker may seek to exhaust the limited energy sources of the legitimately nodes by intentionally causing collisions. The flow of information may also be suppressed by jamming the wireless channel. |
| | Selective Forwarding | A malicious node forwards only a fraction of the data it is requested to. |
| | Spoofing and Tampering | An attacker may falsify/modify packet data, source information, or acknowledgements. |

Once an attacker's malicious nodes are actively participating in a network, he or she may then begin attempting actions intended to reach the ultimate goal of the attack. In networks that propagate confidential information, Information Theft attacks may

not physically impact network performance, but may present severe consequences from a security perspective. On the other hand, denial of service (DoS) attacks may be used to render the network unusable, removing a source of information in a greater situation awareness system. Selective Forwarding, Spoofing and Tampering attacks generally are not intended to hinder network performance, but rather, to poison or bias the stream of information being served by the WSN. The latter class of attacks involves the falsification of data or its origins in a manner that would benefit the attacker. Selective forwarding attacks, on the other hand, refer to attackers dropping a fraction of data messages they are requested to forward. Such an attack may be intended to simply limit the rate at which information about a monitored environment is received, or to drop specific types of data in order to bias the perception of monitored events. Note that even in cases where the confidentiality or authenticity of data can be assured, selective forwarding is still a threat.

Selective forwarding attacks prove to be particularly interesting in WSNs due to the existing challenges in communicating over a wireless channel. Often, it is difficult to differentiate between messages dropped due to interference or malicious intent. Other data-centric attacks such as tampering may prove more difficult for an attacker to hide in cases where data corruption/modification is not a typical failure mode. Because of the challenges associated with selective forwarding attacks, we have chosen to study and address this particular attack class in our work. While an attack-reactive scheme may ideally provide resilience to attack classes across the varying stages of an attack, reacting to attacks in the *End Goal* stage is still valuable because it is in this stage that an attacker begins attempting to damage the network or the validity of information handled by the network.

## 2.2 Countermeasures

Existing work in the field of WSN security may be generalized into categories of prevention, detection, multi-path resilience, and reaction mechanisms. While significant work has been done in the first three categories, there are fewer reactive works and opportunities for further improvements.

Preventative mechanisms seek to limit an attacker's ability to eavesdrop on, join, or participate in the network. This is typically achieved through authentication, or by designing protocols to reduce the opportunity for specific attacks to be mounted.

LEAP+ introduced methods by which cryptographic keys could be established between nodes and base stations, between pairs of nodes, between clusters of nodes, and between all nodes in the network [11]. SPINS provides protocols for achieving data confidentiality, broadcast authentication, and assurance of data freshness [24]. The work of Ning, Liu, and Du introduced DoS resilience into a broadcast authentication scheme, through the use of a mechanism referred to as "message-specific puzzles" [12]. The common elements in works on authentication in WSNs are generally "weak" authenticators based upon chains of hashing functions. While modern cryptographic schemes are believed to be too inefficient on the microcontrollers typically used in WSN node platforms, hash chains have been shown to be easy to verify by nodes, but computationally difficult to forge. Despite public key cryptography often being regarded as too inefficient for use in WSNs, Liu and Ning have illustrated in both theory and implementation the feasibility of using public key cryptography for confidentiality and authentication purposes, via elliptic curve cryptography [13].

As previously noted, the physical capture of network nodes and the extraction of network secrets is commonly regarded as a realistic threat, requiring additional layers of preventative security, which often are intended to counter specific threats. For example, INSENS, a multi-path routing scheme designed for attack resilience includes an "echo-back" scheme intended to prevent successful attempts at carrying out a rushing attack, in which an attacking node advertises route availability with a very high transmission power, tricking downstream nodes into attempting to use a non-existent link [25]. INSENS's "echo-back" mechanisms allows nodes to verify the bidirectional property of a link, thereby thwarting such an attack. Another example of an attack-specific countermeasure is the concept of a packet-leash, introduced in [9], designed to defend against wormhole attacks, in which malicious nodes tunnel messages to different areas of a network through an out-of-band low latency link. The packet-leash solutions presented by Hu, Perrig, and Johnson effectively limit the geographic distance packets are able to travel, as well as the rate they can travel [9].

Works on the detection of WSN attacks provide another layer of security, providing situational awareness with respect to malicious actions in the network. These works generally focus on identifying anomalous behavior that is indicative of an attack. In their survey paper, Rajasegarar, Leckie and Palaniswami describe machine learning-based approaches used in the field, including data clustering, support vector machines models, and population density models [17]. These techniques generally require the

use of specialized watchdog nodes in the network, due to the computational complexity of the machine learning algorithms. For schemes intended to be run on the "normal" network nodes, rule-based schemes have been explored. For example, [19] presents a distributed scheme in which nodes monitor properties such as transmission intervals, delays, repetition and integrity and compare these against known "good" values in order to identify suspicious behavior. For the detection of selective forwarding attacks specifically, [26] and [27] propose that gateways or checkpoint nodes propagate reception acknowledgements back to nodes generating data; a failure to receive such an acknowledgment is indicative of a selective forwarder in the routing path. Our reactive scheme presented in this work leverages the Retransmission Rule presented in [19], which seeks to determine whether or not a node has retransmitted a message it was requested to forward. The choice of detection strategy was based upon the notion that the Retransmission Rule evaluation could be performed passively by individual nodes, whereas the schemes presented in [26] and [27] require the downstream propagation of additional acknowledgement traffic.

Another important focus in WSN security is attack-resilience; networks are designed to operate in the presence of attackers. A significant number of works have investigated the use of multipath routing to achieve attack resilience. Originally designed for fault-tolerance, multipath routing duplicates data along multiple braided or disjoint paths in order to increase the likelihood of the data reaching a sink node. Protocols such as INSENS [25], SEEM [28], and H-SPREAD [29] have augmented the multipath design philosophy with authentication, key establishment schemes, and preventative measures. A thorough analysis of numerous secure multipath protocols can be found in [21].

In reacting to attacks, this also work attempts to duplicate data messages along duplicate paths. Unlike multipath routing schemes however, an end-to-end secondary paths are not created and maintained as part of normal network operation. Instead, a major goal of this work is in evaluating the effectiveness of simply choosing a secondary forwarder on-the-fly when attacks are detected.

While numerous works have explore prevention, detection and multipath resilience, far fewer efforts have been made to explore how the aforementioned works can be combined to allow WSNs to automatically react to ongoing attacks in a manner that will minimize the impact of attacks on network operations. In response to detected attacks, the scheme presented in [23] reclusters a network to exclude malicious nodes, and attempts to recover compromised nodes via over-the-air programming.

The SARP protocol, detailed in [22], utilizes multi-dimensional trust metrics to allow nodes to evaluate and report the trustworthiness of neighboring nodes, and then adapt their routes to avoid nodes deemed untrustworthy.

While both of these schemes allow networks to effectively react to attacks in an automated fashion, they require the use of auxiliary communication and coordination to facilitate reactive actions; this implies some additional overhead and attack surfaces. This work seeks to identify if a reactive scheme can be achieved with nodes making reactive decisions independently, or with minimal auxiliary information exchanged between nodes.

## 2.3 Collection Tree Protocol

To facilitate the design of a distributed and reactive routing scheme, the Collection Tree Protocol (CTP) has been selected as a baseline protocol. CTP is included in the the TinyOS[1] operating system and protocol suite for low power sensor networks. This particular protocol has been selected for its emphasis on reliability, efficiency, adaptiveness with respect to varying link qualities, and availability in both simulation and real-world implementations.

As reported by the authors in [30] and separately verified in [31], CTP is capable of achieving 99.9% data delivery rates in networks with sufficient connectivity. One of the major reasons for CTP's success over other tree-based routing protocols is its use of a link-quality based routing gradient, as opposed to a simple hop-count to a root node. This metric, referred to as Estimated Transmissions (ETX), provides nodes with an approximation of the number of transmissions required to propagate a message to a sink node, considering the number of retransmissions needed along each link. This metric is evaluated by monitoring both the incoming and outgoing links shared between a node and its neighbors, and summing these estimates along a path.

CTP evaluates outgoing links by maintaining counts of the number of data messages transmitted to a neighboring node, and the number of acknowledgements (ACKs) received from that node. Incoming links are estimated through the use of routing beacons, which nodes use to advertise their current routing gradient information,

[1] `http://www.tinyos.net/`

parent node, congestion status, and to request route updates. Each time a node broadcasts a routing beacon, it increments a local beacon sequence number, which is embedded into the routing beacon frame. By monitoring the gap between current and previous beacon sequence numbers, a node can estimate the number of retransmission attempts required to exchange a message with its neighbor. Both of these incoming and outgoing link statistics are maintained over windows of data and routing beacon messages. At the end of each window, CTP calculates the ETX associated with a neighboring node using a weighted sum of both the current and previous link statistics, which provides exponential smoothing of these estimates. The cost to reach a root node in CTP is calculated by summing the ETX values along each hop.

Sink nodes in CTP advertise a nominal ETX of 0 to denote that they are roots of the routing tree. This effectively yields an "addressless" scheme in which nodes do not require *a priori* knowledge of sink node addresses, and instead discover paths to root nodes in a distributed manner, based upon the fact the routing cost (ETX) should always decrease along a path to a sink. This particular feature of CTP is noteworthy, due to the potential for inherent fault and attack resilience through the addition of root nodes to a network.

Currently, the available implementations of CTP do not provide security mechanisms of any form[2]. In their survey of attacks and countermeasures in WSNs, Karlof and Wagner note that CTP's lack of authentication in route updates allows for malicious nodes to perform a sinkhole attack by advertising a falsely low ETX in their routing beacons [8]. They also note how wormhole attacks can be used to create subtrees in CTP topologies, even if route updates were authenticated, and how malicious nodes can induce routing loops through spoofing attacks.

In addition to the attacks on CTP described in [8], we have also found selective forwarding attacks to be trivial to carry out in CTP. To aid in the description of this attack, a graphical representation of the forwarding path present on each node is presented in Figure 2.1.

[2]Therefore, the design of an attack-reactive scheme based upon CTP is believed to be novel work.

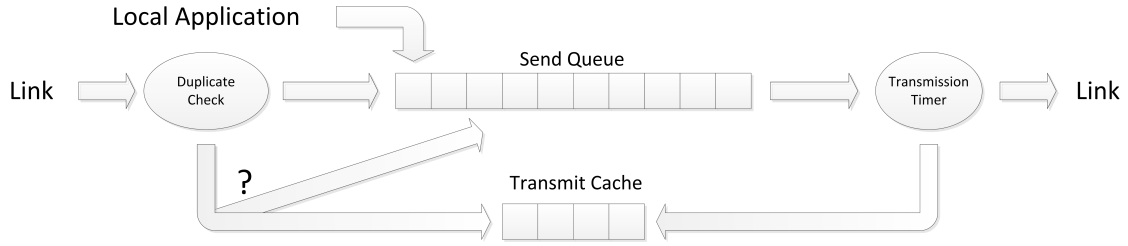


Figure 2.1: CTP data forwarding path

As shown above, messages are added to a node's *Send Queue* when:

- A local application generates new data
- A forwarding request is received from the link layer and a duplicate of the message is not already in the *Send Queue* or in the *Transmit Cache* of previously sent messages.

A timer event is used to determine when to dequeue a message and attempt to transmit it to a node's parent. Once a message has been dequeued, it is periodically retransmitted to the parent node until the link layer signals reception of a MAC-layer acknowledgement, or a retransmission limit is hit. The former case indicates successful reception, whereas the latter is logged as a failure. Note that this scheme only requires acknowledgement that a message has been received by the parent's link layer; no verification of the parent actually forwarding the message is required.

Therefore, to carry out a selective forwarding attack, an attacker only needs to suppress the operation of a message being inserted into the *Send Queue*. In terms of a code injection attack on a mote platform, this may be achieved through the insertion of a conditional branch intended to skip the message enqueue operation. From the perspective of an attack model, this corresponds to a node ACKing a message, but never forwarding it.

# Chapter 3

# Design of Reactive Routing Scheme

This chapter details the design of the reactive routing scheme for selective forwarding attack resilience. As noted in the previous chapter, TinyOS's CTP is used as a foundation for this scheme due to its high data delivery goals, link-quality based routing gradient, and inherent multi-gateway support. The designed mechanism consists of two main modifications to CTP: a Parent Reliability Estimator (PRE) and changes to the route update procedure. The PRE allows nodes to estimate whether a parent node may be failing or maliciously dropping data messages, and categorize the parent as being *Good*, *Suspicious*, or *Unreliable*. Based upon this categorization, the modified route update procedure is used to duplicate data through a secondary parent node, or immediately blacklist the current parent and choose another. Figure 3.1 presents a high-level flowchart describing the design of our reactive routing scheme, which is described in the following sections.

This chapter first establishes the scope of this work via a discussion of the threat model and assumptions about attackers. Next, the PRE mechanism and its associated metric are detailed. Finally, the modified route update procedure, which utilizes the PRE metric for reactive decision-making, is presented.

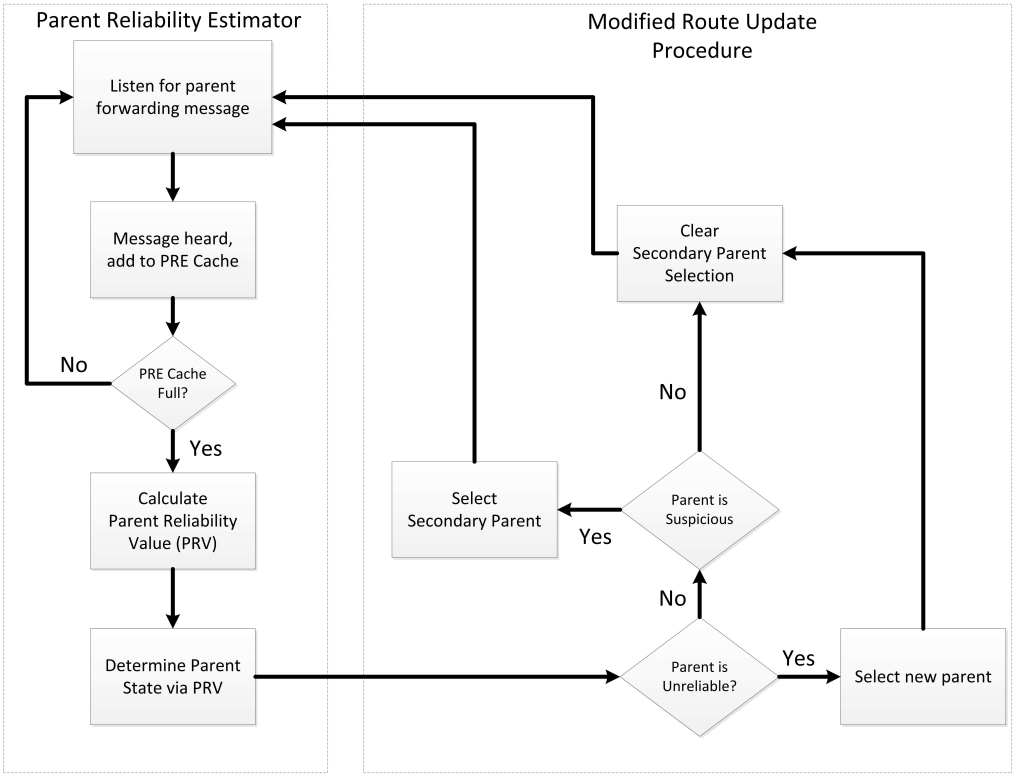


Figure 3.1: Reactive routing scheme flowchart

## 3.1 Threat Model and Assumptions

Our work focuses specifically upon selective forwarding attacks, in which an attacker drops a fraction of all data messages it is requested to forward. As noted in the following section, this work may easily be extended to also detect and react to data tampering attacks, in which an attacker modifies the payload of a message before forwarding it. However, the evaluation of data tampering resilience is left as a future work.

The scope of attacker actions addressed in this work is limited to selective forwarding attacks only; it is assumed that the attacker will not modify, suppress, or spoof CTP beacon message contents. Such attacks may be used to attack routes toward the attacker, which may be considered a form of sinkhole attack. Instead of preventing such an attack from occurring, this mechanism seeks to react when data-level attacks begin. Being that there are few, if any, works that apply such protections to CTP specifically, this work leaves this aspect open to future investigation. Existing work regarding the use of Elliptic Curve Cryptography (in TinyOS) may provide one

possible avenue for introducing preventative defenses into CTP [12].

In order to simplify simulation models, attacking nodes will be assumed to randomly (with a uniform distribution) drop a static, fixed percentage of data messages they are requested to forward. Although a real-world attacker would be likely to drop data based upon its content, origin, or relation to ongoing events, this simplified model is intended to be a first step in determining if the designed reactive routing mechanism is capable of yielding improvements.

As noted in Section 2.3, CTP utilizes a MAC-layer acknowledgement to determine when a parent node has successfully received a data message; there is an inherent assumption that the parent node will correctly forward the message, given that its forwarding buffer has not overfilled. A violation of this assumption proves to be an effective means to carry out a selective forwarding attack on the CTP protocol. By responding with the MAC-layer ACK, regardless of whether or not it actually forwards the data, a malicious node can attempt to maintain a high data-based ETX (as calculated by its child). Through its data-based ETX calculations, CTP provides inherent resilience to selective forwarding attacks that suppress MAC-layer ACKs; after repeated un-ACKed transmission attempts, the victim child node will eventually deduce that its has a low quality link to its parent and seek to chose a different parent node. Because of this inherent resilience, this work's selective forwarding attack model focuses on the case which CTP is vulnerable: when an attacker that does not suppress MAC-layer acknowledgement of received data messages.

## 3.2 Parent Reliability Estimator

### 3.2.1 Design Overview

One of CTP's major improvements over other tree-routing schemes is its use of a routing gradient that considers link quality, rather than hop counts alone. This improvement is achieved through CTP's usage of the 4-bit link estimator [30]. This work seeks to further extend the success of this design decision by also considering nodes' data forwarding reliability into CTP's parent selection procedure. The mechanism by which this work monitors a parent node and quantifies its forwarding reliability is referred to as a Parent Reliability Estimator (PRE).

The PRE is designed to address a selective forwarding attack (or failure) than can

arise if a CTP node replies to a data message with a MAC-layer ACK, but does not actually retransmit the packet. Per the retransmission rule presented in [19], the PRE listens for the parent's retransmissions of data messages that have been previously sent. Unlike [19], this retransmission rule is evaluated by each node individually, instead of by network monitors and neighboring nodes. By monitoring the number of messages a parent correctly forwards over time, a node's PRE labels the current parent as being *Normal*, *Suspicious*, or *Unreliable*. When the parent's PRE state is *Normal*, a node assumes that no failures or attacks are associated with its parent, and operates per the baseline CTP specification. An *Unreliable* parent is one that has repeatedly failed to forward a significant amount of data and cannot be trusted; a node blacklists this parent node and selects a different parent. A state of *Suspicious* is used to indicate to the routing protocol that preliminary action should be taken for a possibly failing or attacking parent. The purpose of this middle state is to allow a node to react before significant data loss occurs, but in a way that will not significantly hinder network operations should the state be asserted as a result of a false positive. The reactive actions associated with a *Suspicious* parent consist of selecting a secondary parent and addressing data messages to both the primary and secondary parents. The remainder of this section discusses the design of the PRE. The following section details secondary parent selection and usage, as well as the associated route update process.

### 3.2.2 Changes to the CTP Forwarding Path

In order to facilitate the monitoring of a parent node, an additional message cache has been introduced into the CTP forwarding path, as shown in Figure 3.2. Conceptually, this additional cache is needed because the existing Transmit Cache is designed to be small, and is only concerned with determining if a message has been previously sent (to any parent). Therefore, we need a slightly larger cache that tracks messages sent to a specific parent. These caches are shown as separate entities in Figure 3.2, but could certainly be implemented in a manner that re-uses common data structures to conserve memory.

As messages are removed from the CTP Send Queue due to reception of a MAC-layer ACK, message data is inserted into both the existing Transmit Cache and a new PRE Cache. The determination of which message fields are copied into the PRE cache would be dictated by implementation-specific memory requirements. At a minimum,

an implementation could store only the data frame fields required to identify the data message, such as the origin and sequence number (3 bytes). However, to additionally ensure the integrity of messages retransmitted by a parent, the storage of a message authentication code (MAC) is also recommended[1]. Note that an implementation capable of verifying the integrity of a retransmitted message could potentially detect and react to data tampering attacks. As the scope of this work is limited to selective forwarding attacks, this will not be addressed further here.

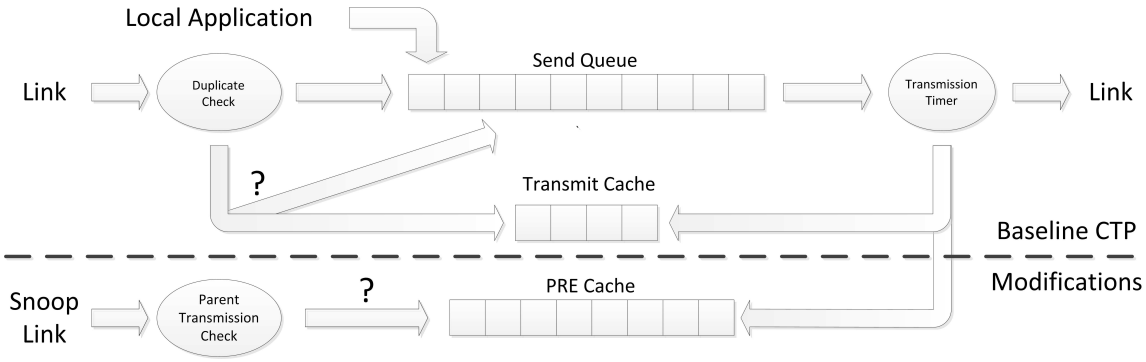


Figure 3.2: CTP forwarding path with PRE cache

### 3.2.3 Calculation of PRE Metric

As ACK'd messages are copied into the PRE cache, the TinyOS snoop interface is used to overhear messages sent by a node's parent and mark the corresponding entries in the PRE cache (see Appendix A). This process is carried out over a window of $w_{PRE}$ messages. At the end of this window, a node computes the current Parent Reliability Value (PRV) $\rho_i$, using number of messages sent to the parent ($n_s$), the number of messages forwarded by the parent ($n_f$), and the previous PRV ($\rho_{i-1}$). The evaluation of $\rho_i$ is shown in (3.1). $\rho_0$ is initialized to a nominal value of 1.0 when nodes are added to the CTP routing table[2].

[1]CTP does not currently compute and include MACs into message frames; this functionality, as described in [11] and [24], would first need to be integrated into to CTP.

[2]Given the limited number of entires in the CTP routing table, this initialization may be abused by an attacker that has accumulated an unfavorable PRV. By halting its routing beacons, an attacking node can attempt to be removed from a victim node's routing table. Once removed from the routing table, the attacker can resume beaconing to be re-added to routing tables with $\rho_i$ initialized to 1.0. Therefore, it is recommended that entries removed from the CTP routing table have their PRVs stored in memory in order to be retrieved upon re-insertion into the table.

$$\rho_i = \alpha_{PRE} \frac{n_s}{n_f} + (1 - \alpha_{PRE})\, \rho_{i-1} \tag{3.1}$$

As done with CTP's ETX calculation, exponential smoothing is achieved through the use of a weighting factor $\alpha_{PRE}$. Due to possibility of noise and collisions in a wireless channel, as well as the likelihood that a node may not have its radio in receive mode at the moment which its parent forwards a data message, a low weighting factor is required; a low value favors the parent's previous forwarding history, and attempts to smooth any transient fluctuations in order to reduce false positive assertions of a *Suspicious* or *Unreliable* state.

Given that the PRV is evaluated over windows of messages, it follows that the PRE's response time in detecting a failing or attacking node is a function of $w_{PRE}$, $\alpha_{PRE}$, and the rate at which a node sends messages to its parent. To achieve an initial estimate of response times and the PRE's detection capabilities, the PRVs associated with varying attackers and are plotted for $\alpha_{PRE} = 0.1$ and $\alpha_{PRE} = 0.25$ in Figures 3.3 and 3.4, respectively. A window limit of $w_{PRE} = 10$ is used for both of these cases, as well as in simulations.

As shown in the plot for $\alpha_{PRE} = 0.1$, the PRVs for attackers dropping 50% or more data quickly rise above 1.5 within 75 messages. Detection of a 35% attacker may be possible after approximately 200 messages. Unfortunately, attack rates below 35% do not appear to significantly raise the PRV value, and may not be differentiable from channel-related failures or missed transmissions due to a node's radio not being in the receive state.

The $\alpha_{PRE} = 0.25$ plot shows that this choice in weighting factor results in a much faster response. Only 30 messages are needed for the PRV of attackers dropping more than 50% of their data to reach approximately 1.5. A 35% attacker reaches this PRV with 110 messages. Similar to the $\alpha_{PRE} = 0.1$ case, the PRVs associated with attackers below a drop rate of 35% converge to values that are likely too low to differentiate with "noise," (albeit, at a quicker rate).

Based upon Figures 3.3 and 3.4, $\alpha_{PRE} = 0.25$ appears to be a better choice in the sense that it would allow for a quicker assertion of *Suspicious* or *Unreliable* statuses for malicious or failing nodes. As previously noted, however, channel collisions and missed transmissions may result in transient spikes in PRV values.

In order to obtain insight into PRV variability with respect to a choice in $\alpha_{PRE}$, we

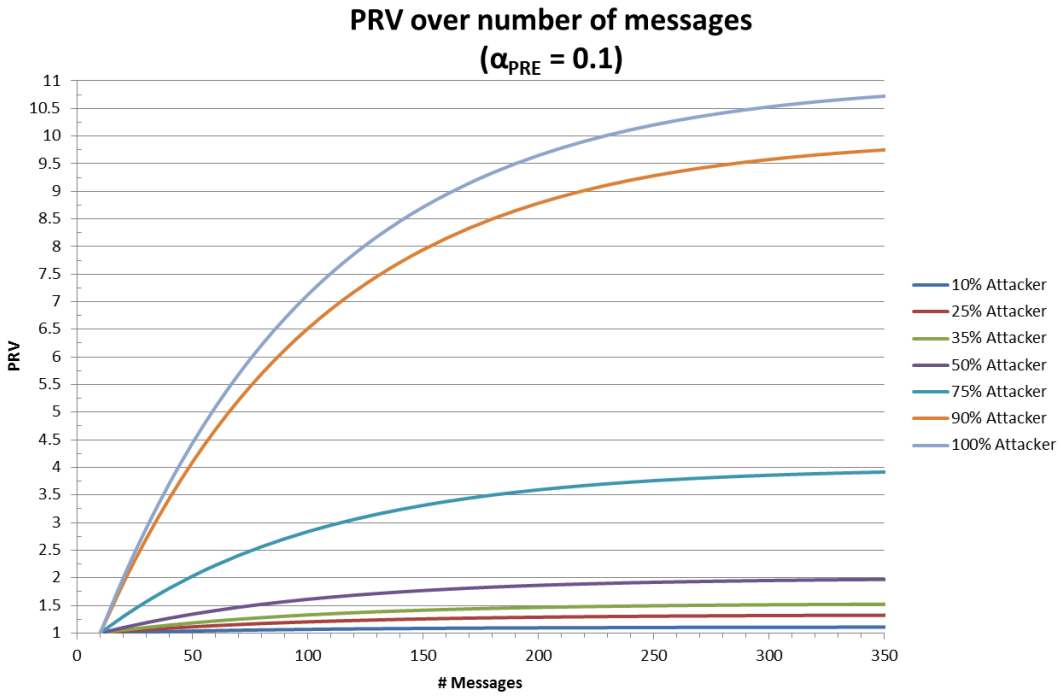


Figure 3.3: PRVs over forwarded messages, for $\alpha_{PRE} = 0.1$

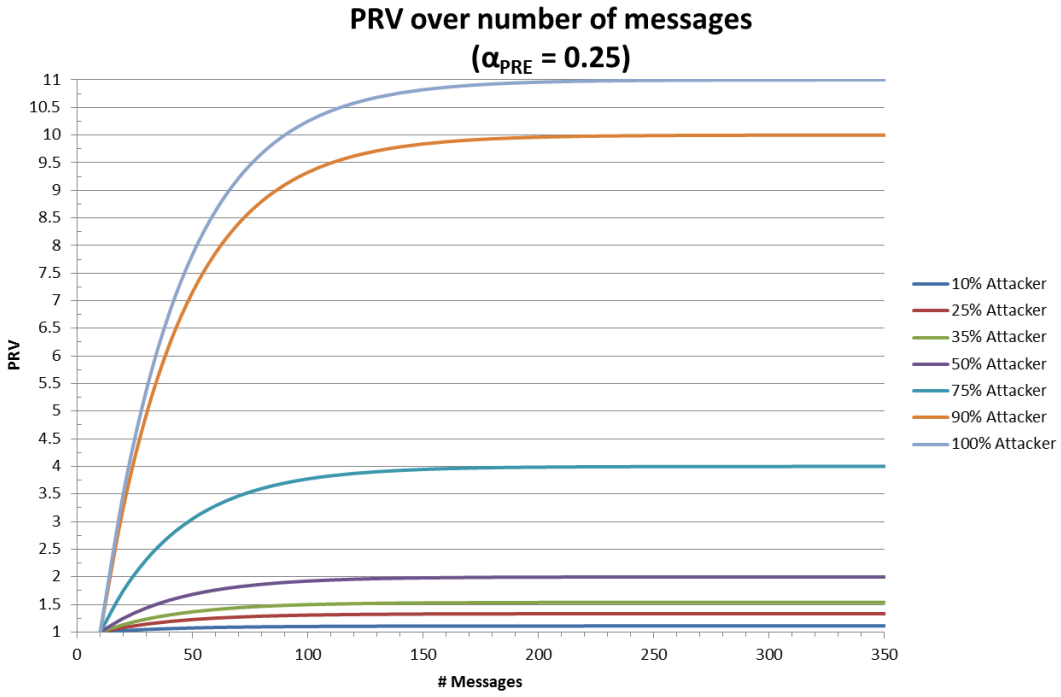


Figure 3.4: PRVs over forwarded messages, for $\alpha_{PRE} = 0.25$

attempt to create a very simple model the aforementioned variations via a random uniform distribution of 10,000 $\frac{n_s}{n_f}$ values in the range [1.00, 1.55]. These random $\frac{n_s}{n_s}$ values are grouped into windows of 10 "messages" in order to calculate an associated PRV. This effectively yields a sample of 1,000 PRV's. This process is repeated for $\alpha_{PRE} = \{0.10, 0.15, 0.20, 0.25\}$, and a boxplot is created for each case. As shown in Figure 3.5, these variations noticeably increase with $\alpha_{PRE}$.

The plots' upper "whiskers" provide approximations of the lowest *Suspicious* state PRV threshold that may be used without incurring too many false positives. This upper bound on PRV variations would likely increase further due to missed retransmissions for entire windows during periods of congestion and frequent collisions. It is recommended that a lower $\alpha_{PRE}$ value, such as 0.10 or 0.15, is used to reduce false positives. Ultimately, the choice of this parameter for a particular application requires a thorough analysis of target operating environment and network configuration. As such, the optimal $\alpha_{PRE}$ is not determined here; a value of 0.10 is used in simulations to test slower, but more effective responses.

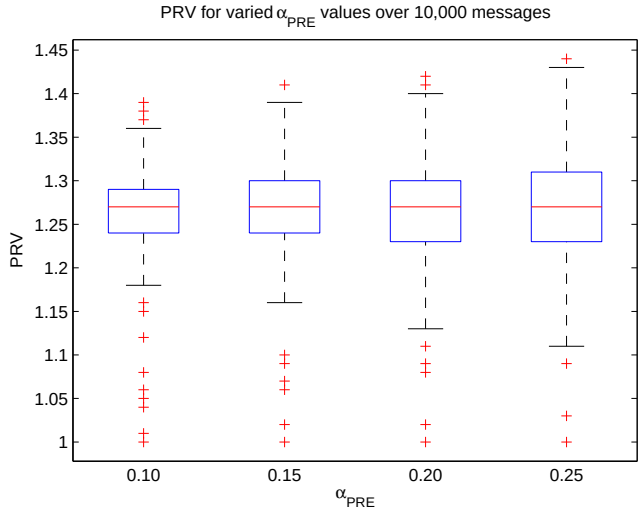


Figure 3.5: PRVs over 10,000 forwarded messages, for various $\alpha_{PRE}$ values

## 3.3 Reactive Routing Decisions

Armed with a quantification of a parent node's forwarding reliability, a node can change it's routing behavior in a manner that increases the likelihood that its data

will reach a gateway. A node's routing decisions are based upon its parents state, as deduced by $\rho$, and two thresholds, $T_{Suspicious}$ and $T_{Unreliable}$. The association between the parent state and $\rho$ is shown in (3.2). The following subsection describes the use of these values and how they relate to a modified version of the CTP route update process.

$$\text{Parent State} = \begin{cases} \text{Normal} & 1.0 \leq \rho < T_{Suspicious} \\ \text{Suspicious} & T_{Suspicious} \leq \rho < T_{Unreliable} \\ \text{Unreliable} & T_{Unreliable} \leq \rho < \infty \end{cases} \tag{3.2}$$

### 3.3.1 Secondary Parent

When a route update occurs, a node's PRV is checked to determine if the parent is marked *Suspicious*. If so, the node chooses a secondary parent using the procedure described in the following section and reports the secondary parent's address in future data frames. To facilitate this, 2 bytes have been appended to the end of the standard CTP data and routing frames, as shown in Figure 3.6. The node continues unicasting messages toward the suspicious parent, but does not explicitly unicast duplicate messages to the secondary parent. This is done to reduce both network transmission overhead and complexity; if a node were to unicast the message to two parents, additional processing would be required to manage queuing and retransmission policies for both parents. A simple implementation might remove a message from the CTP Send Queue after first sending to the primary parent and then to the secondary parent; such an implementation would likely suffer from decreased throughput. Instead, the TinyOS Snoop interface is again used to facilitate the reception of messages by a secondary parent. As shown in the pseudocode presented in Appendix A, upon detecting that its address has been set in the *Secondary Parent* field of a data frame, a node will process the message as if it had been received via unicast.

While the use of the Snoop Interface alleviates the need to retransmit a message to a second node, it introduces the risk of the secondary parent not receiving the message. In CTP, a node will attempt 30 retransmissions to a parent before asserting a send failure[3]. This is expected to be sufficient with respect to wireless channel conditions;

[3]This behavior is defined by *MAX_RETRIES* in CtpForwardingEngine.h. If a radio does not support acknowledgements, a message is only sent once, without any retransmissions. This information was obtained from the current TinyOS version at the time of writing (i.e., version 2.1).

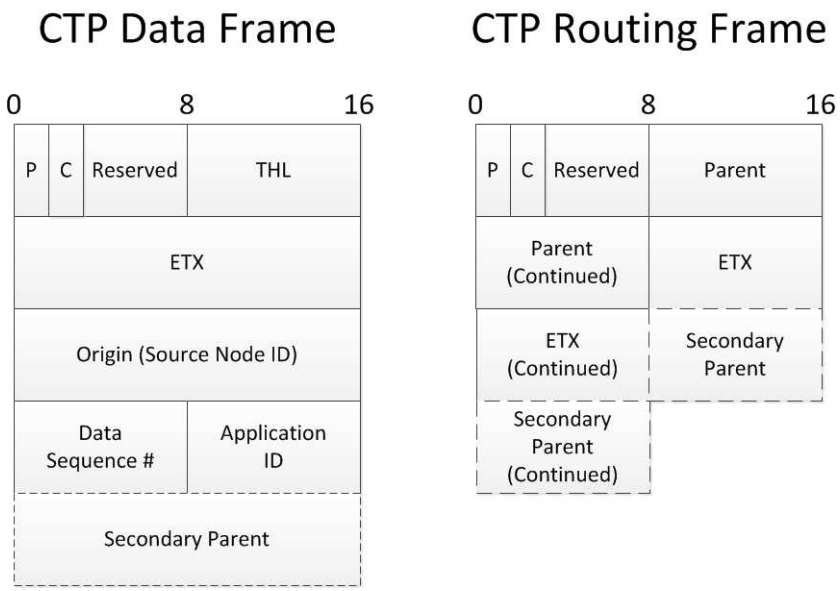


Figure 3.6: CTP frames with added secondary parent field

if the channel conditions alone prevent a node's secondary parent from receiving a message, it is certainly possible the conditions also prevented successful reception by the node's parent. As a result, retransmissions to the node's parent may help ensure the secondary parent receives the message. The secondary parent's radio state is another major factor that may contribute to failures in this scheme. Because nodes do note reply with ACKs to snooped messages, a node has no means to determine if the secondary parent received a message[4].

Keeping in mind that the *Suspicious* state is intended to be a transient state between the assertion of the *Normal* and *Unreliable* states, we believe that this opportunistic approach to utilizing a secondary parent is congruent with the design goals of CTP and our reactive mechanism. During a period when a node is not able to determine if its parent is failing or attacking, the secondary parent functionality is intended to ensure that at least some data is received. If the assertion of the *Suspicious* state is the result of a false positive, then only some additional transmission overhead and data duplication have been incurred. On the other hand, if the *Unreliable* state is eventually asserted, some portion of dropped data may have been recovered. Although

[4]Technically, the Snoop Interface can be used to listen for the secondary parent's retransmission, as done with the parent monitoring in this work. This is not done here in order to reduce complexity and "bookkeeping" requirements associated with the reactive scheme.

not all of the dropped data will have been recovered, ideally, the state of the monitored environment will be possible to infer from recovered data messages.

In the case where a node has not selected or was not able to select a secondary parent, reserved addresses are placed in the associated data message field. This is to maintain a consistent size of the data frame, as well as to support the reactive routing decisions detailed in the following section. Table 3.1 presents the reserved addresses in this work's modification of CTP and the meaning of each address.

Table 3.1: Reserved addresses in reactive routing scheme

| **Reserved Address** | **Value** | **Description** |
|---|---|---|
| *INVALID_ADDR* | 65535 | Used in *Destination Address* field by CTP to denote node has no route. Not used in *Secondary Parent* field. |
| *NO_SECONDARY_NEEDED* | 65534 | Indicates node's PRV does not require selection of a secondary parent. |
| *NO_SECONDARY_FOUND* | 65533 | Indicates node's PRV requires a secondary parent, but no candidate nodes were found. |

### 3.3.2 Modified Route Update Procedure

A modified version of the CTP route update procedure is used to select primary and secondary parents in a manner that will avoid failing or attacking nodes. As noted in the previous section, the assertion of the *Suspicious* status results in node selecting and using a secondary parent at the next scheduled route update process; this delayed response is intended to maintain standard CTP operation during the transition to the *Suspicious* state. The transitionfrom *Suspicious* to *Unreliable*, however, immediately triggers a route update and blacklisting of the current parent.

A preliminary implementation of this work utilized the existing CTP route update procedure, which chooses the "best" parent candidate based upon the ETX metric. However, a number of very high message latencies and THLs were observed. Upon further investigation, it was discovered that the blacklisting and secondary parent functionality was inducing large routing loops, especially when sibling nodes reacted at approximately at the same time; as parent selection is not coordinated between nodes, they would often select one another or their children. Although the baseline CTP functionality eventually resolves loops, it is undesirable to induce any taxing

behavior in the network[5]. Therefore, modifications to the CTP route update process have been made to force better selections for primary and secondary parents.

The modified route update process (pseudocode presented in Appendix B) utilizes not only the 1-hop and path ETX values, but the primary and secondary parent values broadcast in data and routing frames. Based upon these fields and information that can be deduced from information in the routing table, each routing table entry is assigned a "candidate score", as shown in Table 3.2. The route update procedure sorts[6] potential candidate by this candidate score, where a lower value indicates a better candidate. Entries' path ETXs are used as a secondary sorting criteria. Not shown in Table 3.2 is the case where the routing table is a gateway[7], which is associated with a score of 0.

When a node's parent is marked *Normal*, the candidate scoring procedure is not performed, and parent selection is performed using only ETX values. For the *Suspicious* and *Unreliable* states, Table 3.2 establishes the priority of candidates based upon entries' advertised parent and secondary parent fields. Routing table entries that report a (secondary) parent that is a child of the node performing the route update, blacklisted, or invalid (i.e., entry has no route) are considered not to be viable options; these would result in a loop, usage of an untrusted node, or a "dead end," respectively.

When a node's parent is *Suspicious* and a routing table entry is a sibling node, it is only considered viable if it reports having a non-problematic secondary parent, or that no secondary parent is needed. The prioritization of these two options has been selected to favor distrust of the parent in order to err on the side of caution. Again, secondary parent selections that would induce loops or the use of an untrusted node are avoided. A node should never select another node as both its parent or secondary parent; this case is also considered invalid. Lastly, if a node reports "Couldn't Find Secondary," it is implied that the entry also believes the shared parent to be suspicious, but was unable to find a viable secondary parent. This might be indicative of

[5]Recall that in CTP route beaconing rates increase during periods of routing loops and congestion. See [30], [31], [32] for more details.

[6]A sorting operation is not actually recommended due to efficiency concerns. As done in the TinyOS CTP implementation, the best candidate can be identified via a single pass through the routing table using a few temporary variables.

[7]Denoted by an advertised ETX of 0. Ideally, some form of broadcast authentication [12] [33] would be used to prevent the abuse of this special case.

a problematic or leaf section of the network, and thus, the entry is not considered a viable secondary parent.

If the routing table entry is neither problematic nor a direct sibling, prioritization is based upon the entry's secondary parent field. The "best" choices (in order) are cases where the secondary parent is a sink, is not needed, or is another non-problematic node. In the event that such cases are not available, cases with less viable secondary parents will be considered. These cases first favor the use of the suspicious parent and routing loops, as there's a chance that the parent may forward the data, and the data can be kept alive via the loop. The remaining "worst-case" scenarios are considered if the aforementioned cases are not available.

Also shown in Table 3.2, in the event that a parent is marked *Unreliable* and then blacklisted, more stringent conditions are imposed in order to ensure a parent with a viable route is selected. Again, the ideal candidate is a sink node with a good link quality. The next best cases include nodes who report not needing a secondary parent, or using a non-problematic secondary parent. For simplicity, the other "worst-case" scenarios are not considered to be viable options; a node will instead choose to advertise no route with the Pull-bit set, awaiting for a more trustworthy route before continuing its data forwarding.

To comply with the existing CTP design specification, a node may also switch from its suspicious parent to a different node in the event that this second node has a significantly lower ETX value[8]. This functionality is intended to ensure the usage of links with good connectivity. If sufficient memory is available on a mote platform, it is recommended that a node's PRV be saved when no longer using it as a parent, such that it may be referenced if ever switching back to it. Although not covered in Table 3.2, an application may require that a node never switch from a *Normal* parent to *Suspicious* (or *Unreliable*) node. This functionality, mentioned here for completeness, is left as configurable feature. In applications where data loss or tampering may not be detrimental, some memory overhead can be reduced by not storing PRVs of previous parents. On the other hand, this memory trade-off and more strict parent change rule may prove worthwhile in an application that necessitates valid information.

[8]The authors of CTP define "significantly" to mean an ETX 2.0 lower if the current parent is not congested, and 1.5 if it is.

Table 3.2: Routing table candidate scoring

| *Parent Status* | *Entry's Parent* | | | | | *Entry's Secondary Parent* | | | | | | *Candidate Score* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Other* | *My Parent* | *My Child* | *Blacklisted* | *Invalid* | *No Secondary Needed* | *Other* | *My Child* | *My Parent* | *Blacklisted* | *No Secondary Found* | |
| **Suspicious** | | | | | 1 | X | X | X | X | X | X | DNC |
| | | | | 1 | | X | X | X | X | X | X | DNC |
| | | | 1 | | | X | X | X | X | X | X | DNC |
| | | 1 | | | | | | | | | 1 | DNC |
| | | 1 | | | | | | | | 1 | | DNC |
| | | 1 | | | | | | | 1 | | | DNC |
| | | 1 | | | | | | 1 | | | | DNC |
| | | 1 | | | | | 1 | | | | | 7 |
| | | 1 | | | | 1 | | | | | | 8 |
| | 1 | | | | | | | | | | 1 | 6 |
| | 1 | | | | | | | | | 1 | | 5 |
| | 1 | | | | | | | | 1 | | | 3 |
| | 1 | | | | | | | 1 | | | | 4 |
| | 1 | | | | | | 1 | | | | | 2 |
| | 1 | | | | | 1 | | | | | | 1 |
| **Unreliable** | | | | | 1 | X | X | X | X | X | X | DNC |
| | | 1 | | 1 | | X | X | X | X | X | X | DNC |
| | | | 1 | | | X | X | X | X | X | X | DNC |
| | 1 | | | | | | | | | | 1 | DNC |
| | 1 | | | | | | | | 1 | | | DNC |
| | 1 | | | | | | | 1 | | | | DNC |
| | 1 | | | | | | 1 | | | | | 2 |
| | 1 | | | | | 1 | | | | | | 1 |

| **Legend** | |
|---|---|
| X | Don't Care |
| 1 | Condition True |
| DNC | Do Not Consider |
| # | Score. Lower indicates "better" |

# Chapter 4

# Simulation Results and Analysis

## 4.1 Simulation Methodology

To evaluate the effectiveness of the reactive scheme described in the previous chapter, a simulation implementation was developed for Castalia[1], a WSN simulator built on top of the OMNeT++[2] discrete event network simulator framework. The CTP implementation developed and evaluated in [31] was used a starting point for modifications[3]. The performance of this baseline CTP implementation was first evaluated in the simulator, and results similar to those reported in [30] and [31] were obtained, verifying that this model was a valid starting point.

Simulations were performed on a 100 node, 10 x 10 grid topology. Although the number of nodes could have been a variable change with simulation sets, this was found to only achieve an increased load on nodes close to a gateway, and provide an increased amount of data for attackers to drop. The former is a general issue with network scalability, as it pertains to CTP; as network load increases the expected behavior is increased congestion and collisions. As the effect of the reactive scheme upon networks with poor connectivity is investigated in another set of simulations, this was deemed unnecessary to simulate. Furthermore, an increased number of nodes was not found to yield significantly interesting results, given that attackers drop a fixed percentage of data; the same general impact would occur as the attacker dropped approximately the same percentage, albeit of more data. As such, 100 nodes was deemed sufficiently realistic for simulations, considering the sizes of real-world

[1] http://castalia.npc.nicta.com.au/
[2] http://www.omnetpp.org/
[3] http://code.google.com/p/ctp-castalia/

testbeds such as Tutornet[4], Kansei[5], and MoteLab[6]. Given that the random component of Castalia's path loss model varies (deterministically) with each additional simulation repetition, geographical topology changes were also deemed unnecessary; examination of raw simulation logs showed that different paths were used between successive simulations, and bad links occasionally were exercised. In short, different routing tree "topologies" were effectively exercised via changes to link qualities, rather than geographical placement.

In order to emulate realistic channel conditions, Castalia's log-loss path loss model, non-bidirectional links, and additive interference were used for all simulations. Data flow on the simulation networks was modeled via 20 second sampling periods on non-gateway nodes, with a 90% likelihood of each sampling "event" resulting in a node generating a data message. A 10 second initialization period was allowed at the beginning of simulations for nodes to construct initial routes, after which the 20 second data generation period began. A 5 second jitter was added to node startup times in order to avoid nodes being unrealistically synchronized and to avoid all nodes in the network attempting to transmit at the same time. Simulations were carried out over a duration of 3,000 seconds (simulation time), yielding a total of 150 data generation periods. For 150 data generation periods with 99 nodes generating data with 90% probability, approximately 13,365 data messages were expected to be generated in each simulation.

When present in the simulation, selective forwarding attackers began attacking at the first data generation period. As the amount of data passing through attacking nodes is expected to vary with their placement in the network, we expect to see fluctuations in the amount of data attackers are able to drop between various simulation configurations and even successive runs of the same configuration, as link qualities are (deterministically) randomized. The 3,000 second simulation duration was chosen to allow ample time for detection, reaction, and post-reaction stabilization to occur. Based upon Figure 3.3, we see that approximately 350 messages are needed to detect a single 50% attacker, using a PRV thresholds at 2.0. If $\frac{1}{4}$ of the network's traffic flows through the attacker, we expect detection to begin occurring after approximately 16 data generation periods. This leaves approximately 134 remaining data periods for the reactive mechanism to then perform data duplication and blacklisting, as well as

[4] `http://enl.usc.edu/projects/tutornet/`
[5] `http://kansei.cse.ohio-state.edu/KanseiGenie/`
[6] `http://motelab.eecs.harvard.edu/`

to verify that the network to reaches and remains in a steady state.

The general simulation strategy consisted of 30 simulation repetitions per variable change, and each simulation group was divided into three sets: the test with the baseline CTP implementation and no attacks, the same test with the baseline CTP implementation with attacks, and lastly, the reactive CTP scheme with attacks. Data was then extracted from the simulation repetitions in which the introduction of the attacker into the baseline CTP implementation resulted in more than a 5% reduction in the amount of data delivered to the gateway.

The metrics used to evaluate simulation results consisted of data delivery ratios (DDR), data duplication ratios, average message latencies, and the number of transmitted data messages and routing beacons. The DDR metric, shown in (4.1) quantifies the amount of data generated by source nodes that reaches a gateway node. This metric was used in [31] to evaluate the Castalia-CTP implementation and verify the 99% delivery rate reported in [30].

$$\text{DDR} = \frac{\text{\# Unique data messages received at gateways}}{\text{\# Unique data messages generated by nodes}} \tag{4.1}$$

The data duplication ratio metric, shown in (4.2), is intended to capture the degree to which the same data messages are repeatedly received at gateways. Based upon the nature of our reactive scheme, we expect the amount of duplicate data reaching gateways increases, when compared to the baseline CTP implementation. However, we would expect that this increase should be minimal on networks without ongoing attacks; a significant increase in duplicated data may be an indicator of false positive assertions of the *Suspicious* parent state and unneeded use of secondary parents. Furthermore, a significantly large data duplication ratio observed in simulations with ongoing attacks may indicate too large of a gap between the selected values of $T_{Suspicious}$ and $T_{Unreliable}$.

$$\text{Data Duplication Ratio} = \frac{\text{Total \# data messages received at gateways}}{\text{\# Unique messages received at gateways}} - 1 \tag{4.2}$$

The average message latency metric presents an approximation for the amount of time required for messages to traverse the network, from source to sink. High values are generally indicative of the frequent occurrence of routing loops. We expect some

increase in average message latencies when reacting to attacks, as non-optimal routes may be used to avoid attacking/failing nodes.

Lastly, the number of data and beacon transmissions is used to identify additional traffic generated by this scheme. While some additional traffic is expected due to the use of secondary parents and the reseting of CTP's route beaconing timer with induced route updates, a minimal increase is desired.

The results presented in the remainder of this chapter, shown in boxplots, have been formatted using the default MATLAB® settings[7]. Boxplot edges correspond to the $25^{th}$ ($q_1$) and $75^{th}$ ($q_3$) percentiles, and the red bar between these edges represents the median. Plot "whiskers" extend to the last datapoints not considered to be outliers, and outliers are plotted separately as red +'s. Equation (4.3) presents the conditions under which a data point, $p$, is considered an outlier. More information on this can be obtained from the MATLAB® website[8].

$$q_3 + 1.5 * (q_3 - q_1) \ < \ p \ < \ q_1 - 1.5 * (q_3 - q_1) \tag{4.3}$$

## 4.2 Operation Over Decreasing Link Quality

The goal of this simulation group is to determine a spacing in the grid topology that yields satisfactory network connectivity, such that it could be used for successive simulations. Secondly, these simulations are intended to verify the expected behavior of false positive secondary parent usage and blacklisting increasing the rate of network degradation under poor link qualities. In this simulation group, no attacks were included. Instead, the distance between nodes in the grid topology was varied between $\{10, 20, 30, 40\}$ m. Via Castalia's path loss model, the increased distances between nodes yielded decreasingly reliable links.

The DDR results associated with this simulation group are presented in Figure 4.1. In these plots, we see for 10 m spacing, our reactive scheme generally operates around the nominal 99% delivery rate. For a $T_{Suspicious}$ set too low, an outlier may be indicative of congestion induced by frequent false positives and use secondary parents. As we increase the node spacing from 10 m to 20 m, we see little impact on performance

[7] As of Student Version 7.10.0 (R2010a)
[8] `http://www.mathworks.com/help/toolbox/stats/boxplot.html`

for sufficiently high thresholds, but a more severe response to thresholds set too low. At 30 and 40 m spacing, we see that the link qualities become poor enough that CTP begins experiencing low data delivery rates. In these cases, our reactive scheme acts like an unbounded, positive feedback system; as poor links make nodes appear suspicious, child nodes begin duplicating extra traffic, inducing congestion and interference. With few viable neighbors, the act of blacklisting neighboring nodes leaves no paths for data to travel. Evidence of this excessive traffic generation can be seen in Figure 4.2. For completeness, the average message latencies and data duplication associated with this simulation group are also presented in Figures 4.3 and 4.4, respectively. In these plots we see the same general trend in our overheads rapidly increasing with decrease link quality. It is important to note however, that these data points are associated only with messages that reached the gateway; this is why data duplication actually appears to improve in some cases for the larger spacings.

It is clear that in the 10 m spacing case, nodes have numerous options for routing, perhaps too many possible routes to be realistic. On the other hand, a 30 m spacing appears to border on unstable cases for the baseline CTP implementation. Therefore, we select the 20 m spacing to use as the testbed for remaining simulations, as it produces little overhead or performance degradation for carefully selected thresholds.

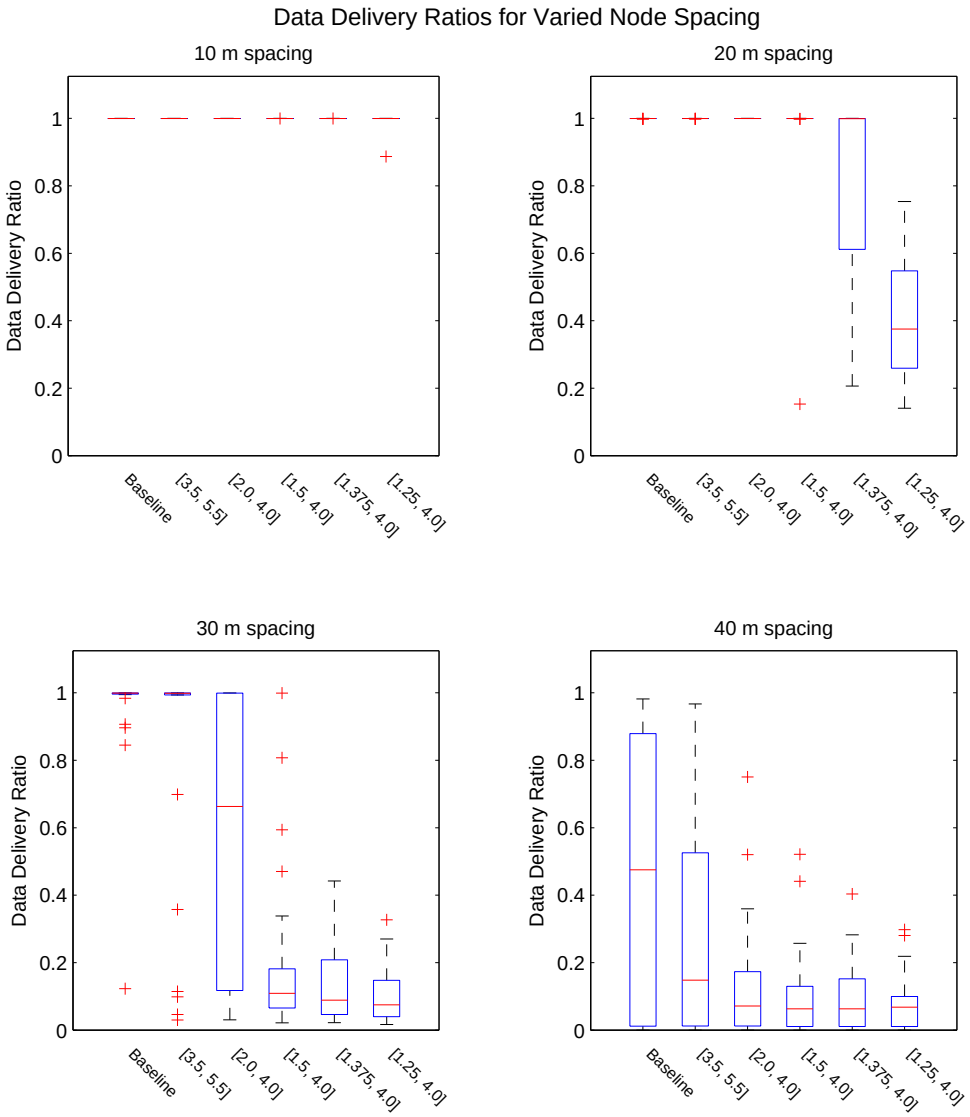


Figure 4.1: Data delivery ratios on networks with varied node spacing

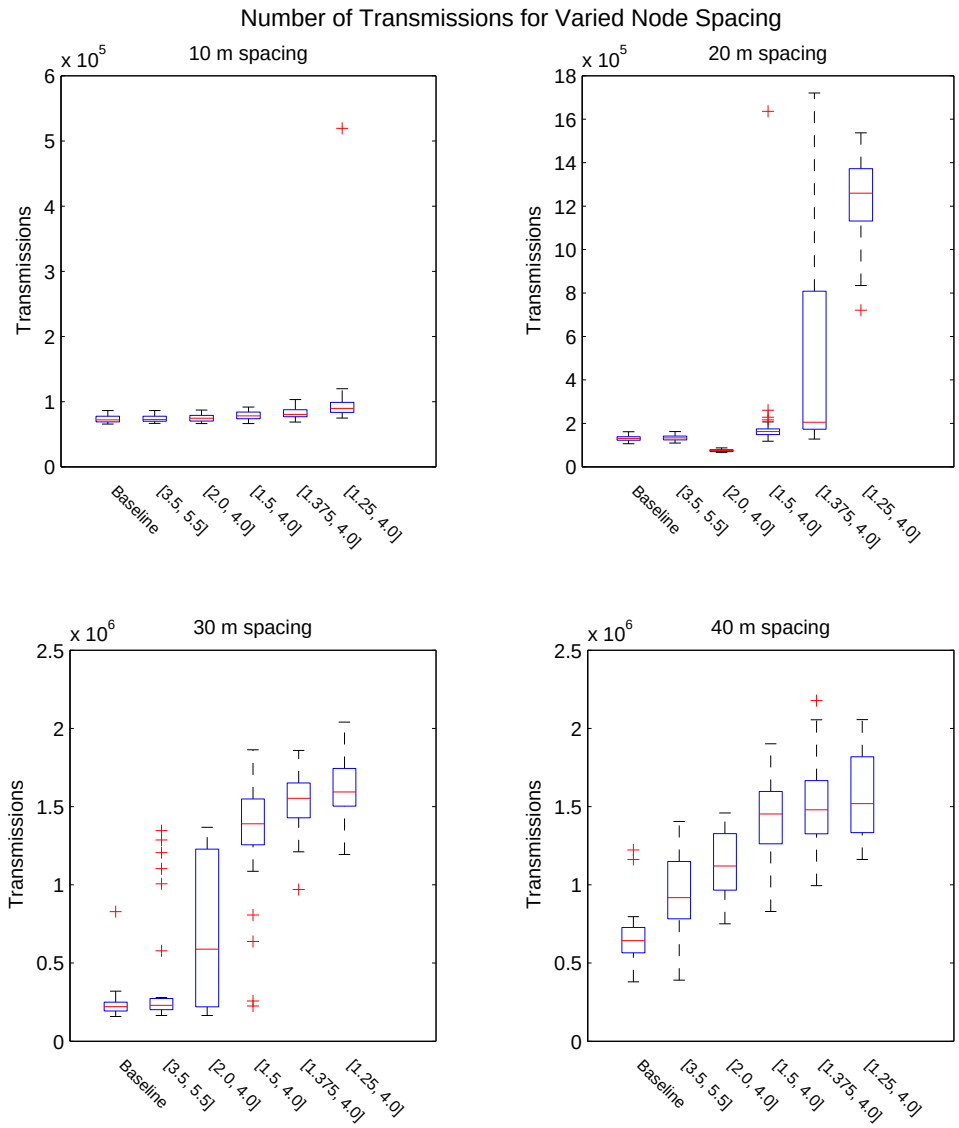


Figure 4.2: Increased secondary parent traffic generated as link qualities decrease

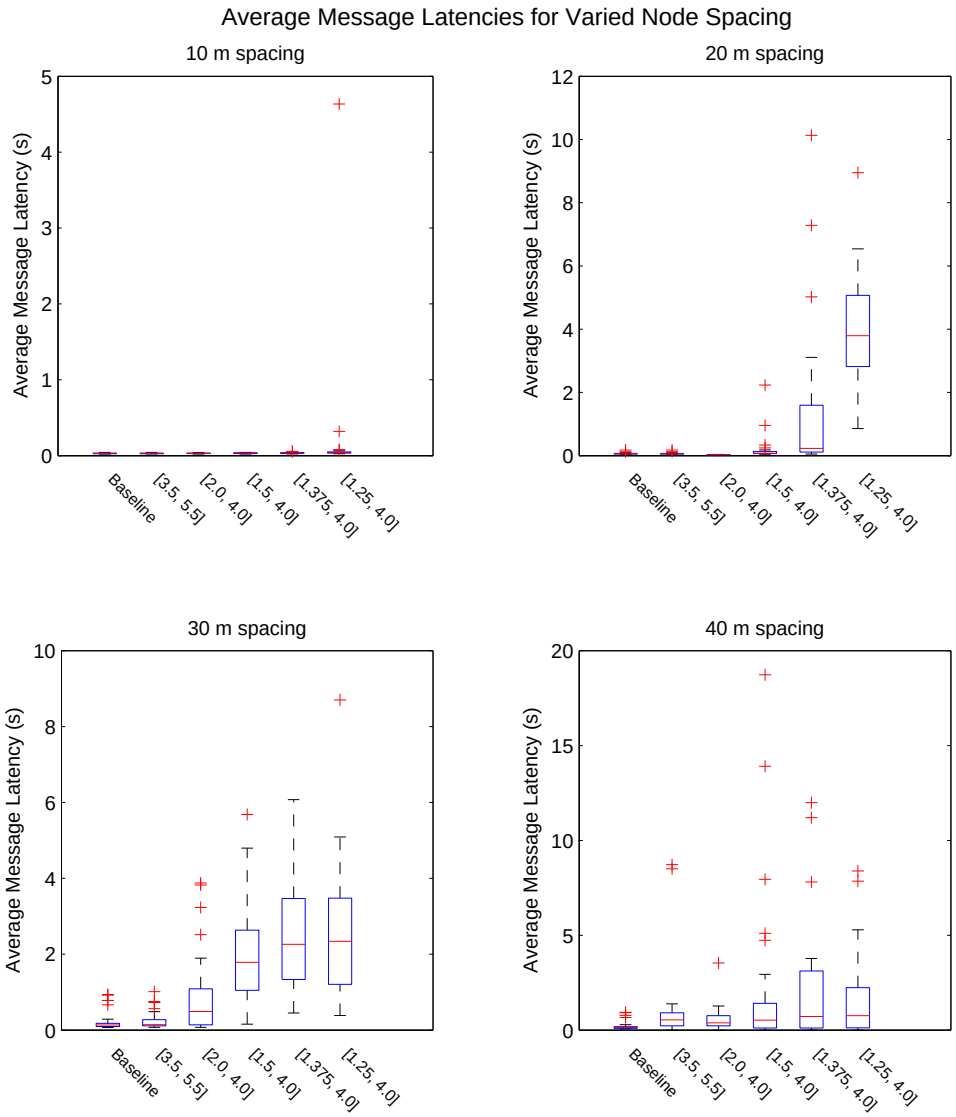


Figure 4.3: Increased message latencies induced by decreasing link qualities

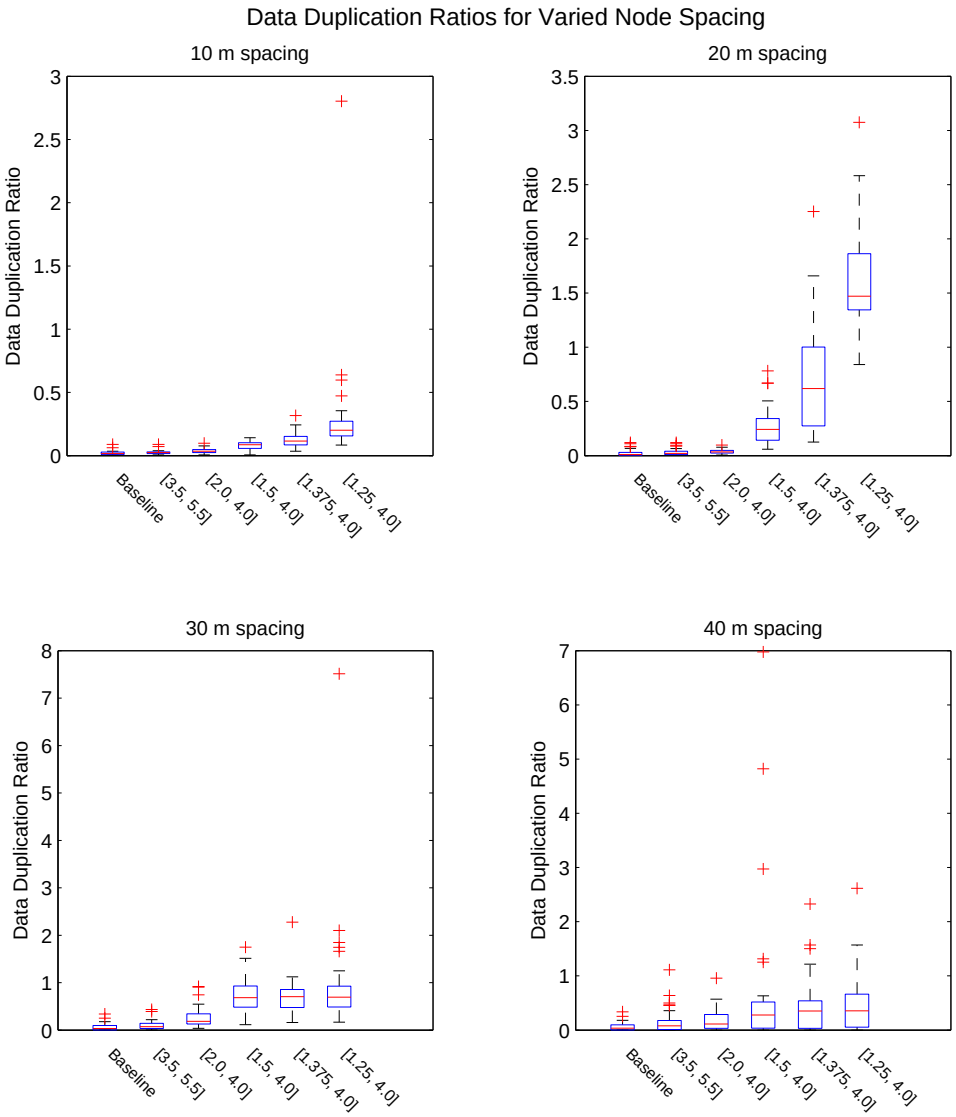


Figure 4.4: Increased data duplication induced by decreasing link qualities

## 4.3 Varying Suspicious and Blacklist Thresholds for a Single 50% Attacker

This second simulation group was selected to investigate the effectiveness against the designed reactive scheme against a single attacker, using various $T_{Suspicious}$ and $T_{Unreliable}$ thresholds on the 20 m spacing grid topology. The attacker was configured to drop 50% of the data it was requested to forward, and was placed near the gateway in order to maximize its ability to adversely affect data delivery. 11 different placements, within 3 hops from the gateway, were tested, with 30 repetitions for each placement. This process was repeated for attacks on the baseline CTP implementation, and then again for attacks on the reactive CTP implementation. A set of simulations run on the baseline CTP implementation with no attacks was used as a reference for comparing overheads induced by our reactive scheme. The results for this simulation group are presented in Figure 4.5. The notation used in the x-axis of these plots is: *NRS* for results that include attacks but do not include the reactive scheme, [$T_{Suspicious}$, $T_{Unreliable}$] for results that include the reactive scheme with the noted thresholds, and *Baseline* for results with no attack and no reactive scheme.

Ideally, a PRV of 2.0 would correspond to only hearing a parent forward 50% of data messages sent to it. Therefore, we would expect setting thresholds at or below 2.0 will allow our mechanism to effectively react to an attacking node, given that it is not the only path to the gateway. The *NRS* boxplot in Figure 4.5's Data Delivery Ratio plot shows that our 50% attacker was generally able to reduce the network's data delivery ratio below 90%. Based upon the Data Delivery Ratio plot in Figure 4.5, we see that the reactive scheme does prove effective for [$T_{Suspicious} = 2.0$ , $T_{Unreliable} = 4.0$] and [$T_{Suspicious} = 2.0$ , $T_{Unreliable} = 3.5$], with the exception of a severe outlier. This outlier is the result of a congested node, being the only viable route 1-hop away from the gateway, becoming blacklisted.

We also see that some data recovery is achieved using higher thresholds, such as [3.0, 4.5] and [2.5, 4.5]. As noted in the discussion of Figure 3.5, we do not expect that a node will overhear all of it's parent's retransmission; for an attacker dropping 50% of received data, we may see PRVs well above 2.0. Therefore, when an attacker's PRV temporarily spikes above 3.0, we see some data recovery from the [3.0, 4.5] and [2.5, 4.5] cases, as secondary parents are put to use. As we lower our thresholds, nodes begin using secondary parents ealier, resulting in much greater data recovery.

When an attacker's PRV spikes above $T_{Unreliable}$ it becomes blacklisted, and future data is routed away from the attacker. Note that in the boxplots that share the same $T_{Suspicious}$ value, but differ in $T_{Unreliable}$ by 0.5, the case with the lower $T_{Unreliable}$ value results in slightly better data recovery, as the attacking node is blacklisted more quickly (if at all).

This plot also indicates a danger in setting thresholds too low. An investigation of the $T_{Suspicious} = 1.5$ outlier cases showed frequent false positive assertions resulted it overuse of the secondary parent functionality, effectively overburdening the network and inducing congestion and interference failures.

The remaining three plots in Figure 4.5 provide insight with respect to overheards introduced by the reactive scheme. In these plots, the reference results are the *Baseline* (no attacks, no reactive scheme). First note that the "best" cases for DDR ($[T_{Suspicious} = 2.0$ , $T_{Unreliable} = 4.0]$ and $[T_{Suspicious} = 2.0$ , $T_{Unreliable} = 3.5]$) did not significantly affect the average message latency of the delivered data, and that only moderate overhead was introduced in terms of data duplication and transmissions. This moderate increase is expected, given our mechanisms strategy of duplicating data via secondary parent nodes. The extreme outlier cases in these box plots were found to correspond to the same 1-hop cases previously mentioned. This indicates that when our reactive scheme begins harming the network, this is observable at the gateways; possible improvements based upon this observation are presented in the next chapter.

Similar to how data recovery slightly improved, we also see that overheads are slightly reduced for smaller threshold windows (i.e. a smaller difference between $T_{Suspicious}$ and $T_{Unreliable}$. This may be attributed to a quicker PRE transition from *Suspicious* to *Unreliable*, which yields less additional secondary parent traffic.

From this first group of simulations, we have verified the effectiveness of our scheme against a single attacker strategically placed near a gateway, identified a range of viable thresholds on this simulation topology, and gained insight into expected overheads.

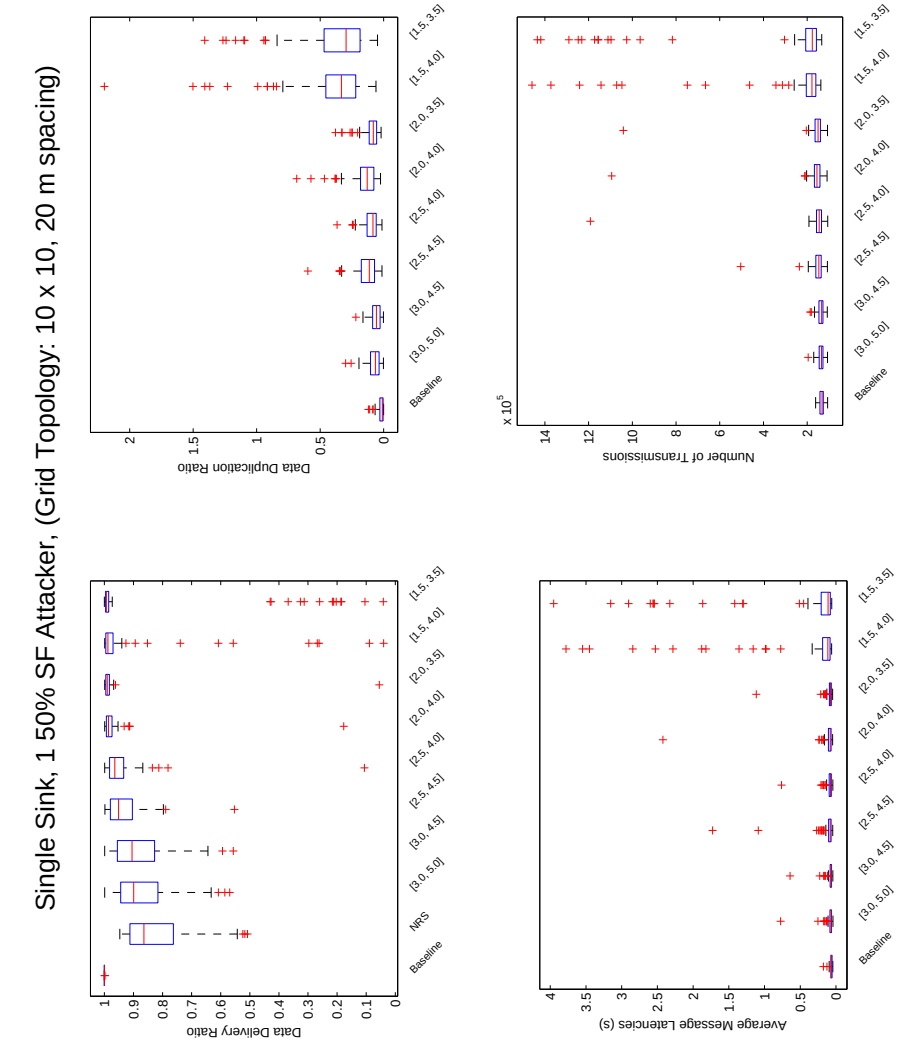


Figure 4.5: Results for varied thresholds against 1 50% drop rate attacker

## 4.4 Varying Attack Rate

After identifying a threshold pair that yielded significant data recovery against a 50% selective forwarding attacker (Section 4.3), the effectiveness of the reactive scheme with this threshold pair was tested against a variety of attack rates. The simulation configuration and attacker placements were the same as those used in Section 4.3. However, these sets were repeated for the following selective forwarding rates: $\{10\%, 25\%, 35\%, 50\%, 75\%, 90\%, 100\%\}$.

As previously discussed, it is expected that an attacker's PRV value will exceed the theoretical value due to channel conditions and unsynchronized radio states. Therefore, the threshold pair used to effectively detect an attacker dropping 50% of messages requested to forward is expected to result in data recovery for slightly smaller drop rates. As shown by the data delivery ratio plots in Figure 4.6, this is indeed the case. For drop rates above 50%, we see significant data recovery (with the exception of the problematic 1-hop cases similar to the ones previously discussed). Despite the threshold being "tuned" for a 50% drop rate, slight data recovery is achieved for the 35% drop case. For lower drop rates, however, we only see limited data recovery; without better baseline connectivity, it is unlikely the thresholds can be set low enough to effectively detect these cases without adversely affecting the network.

These results indicate that our reactive scheme is most effective against attacks and failures that result in large amounts of data loss. While providing little benefit for cases with lower data loss rates, we see that the impact of these lower drop rate attacks is significantly less. However, in some applications, forcing an attacker to drop less data per unit time in order to remain hidden may have benefits. This of course assumes that dropping $X$ amount of data over a longer period of time yields less impact than dropping the same amount of data in a shorter time frame.

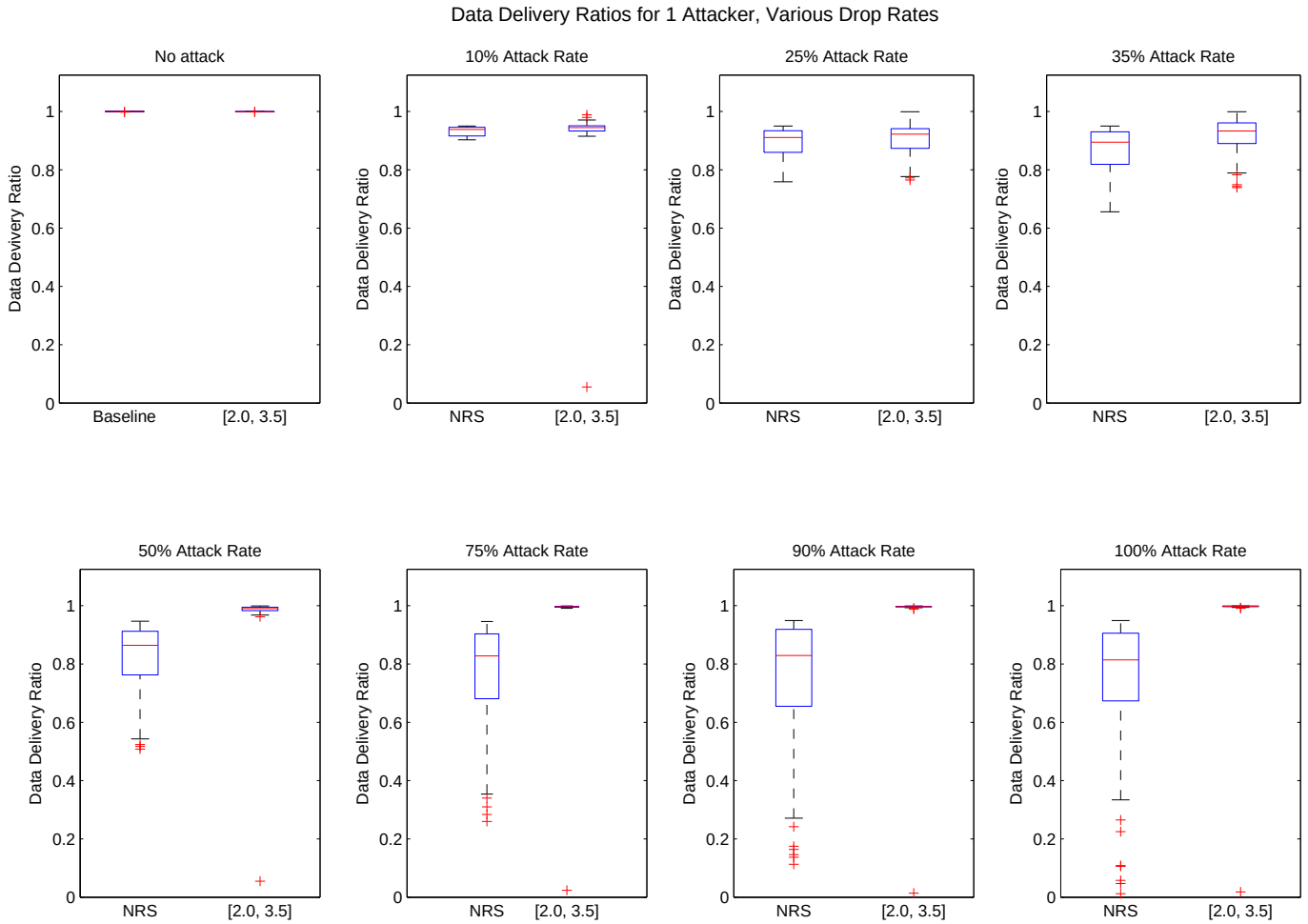


Figure 4.6: Data delivery ratios for a single attacker, various drop rates

## 4.5 Multiple Attackers

In order to evaluate the use of the candidate scoring conditions presented in Table 3.2, another simulation group that includes multiple attackers was used. This simulation group includes a number of attacker placements, presented in the following list. For the random placement cases, configurations were removed if already covered by one of the previous cases. Note that the hop counts presented here are approximate, as they are based upon locations in the grid topology, where a "hop" exists only between adjacent nodes. In all cases, attackers were configured to drop 50% of received data.

- 2 Attackers, 1-hop away from the gateway
- 3 Attackers, 1-hop away from the gateway
- 2 Attackers, 2-hops away from the gateway
- 3 Attackers, 2-hops away from the gateway
- 2 Attackers, 3-hops away from the gateway
- 3 Attackers, 3-hops away from the gateway
- 2 Attackers, randomly placed within 4-hops from the gateway
- 3 Attackers, randomly placed within 4-hops from the gateway

Given that nodes closer to gateways generally have more influence over the flow of data, we expect that the cases where multiple attackers are near the gateway would yield significant impact. On the other hand, the same number of attackers further from the gateway would be expected to be able to drop significantly less data.

Figure 4.7 shows that this expected behavior does indeed occur. For the 1-hop cases, there is often little that can be done to avoid the attackers. In the cases where nodes can transmit directly to the gateway (usually with a less than ideal link), we see significant data recovery. However, the outliers in Figure 4.7 are indicative of cases where the attackers are the only viable routes to the gateway; the blacklisting of attack nodes results in no complete paths to the gateway.

As the multiple attackers are moved further from the gateway, we see that the variance of data delivery ratios decrease, and the median increase, indicating a decreasing impact. As expected, the availability of alternative routing paths increases with the

increased distance from the gateway, allowing for significant data recovery. Furthermore, this verifies our hypothesis that selective forwarding at a particular rate yields greater impact near a gateway.

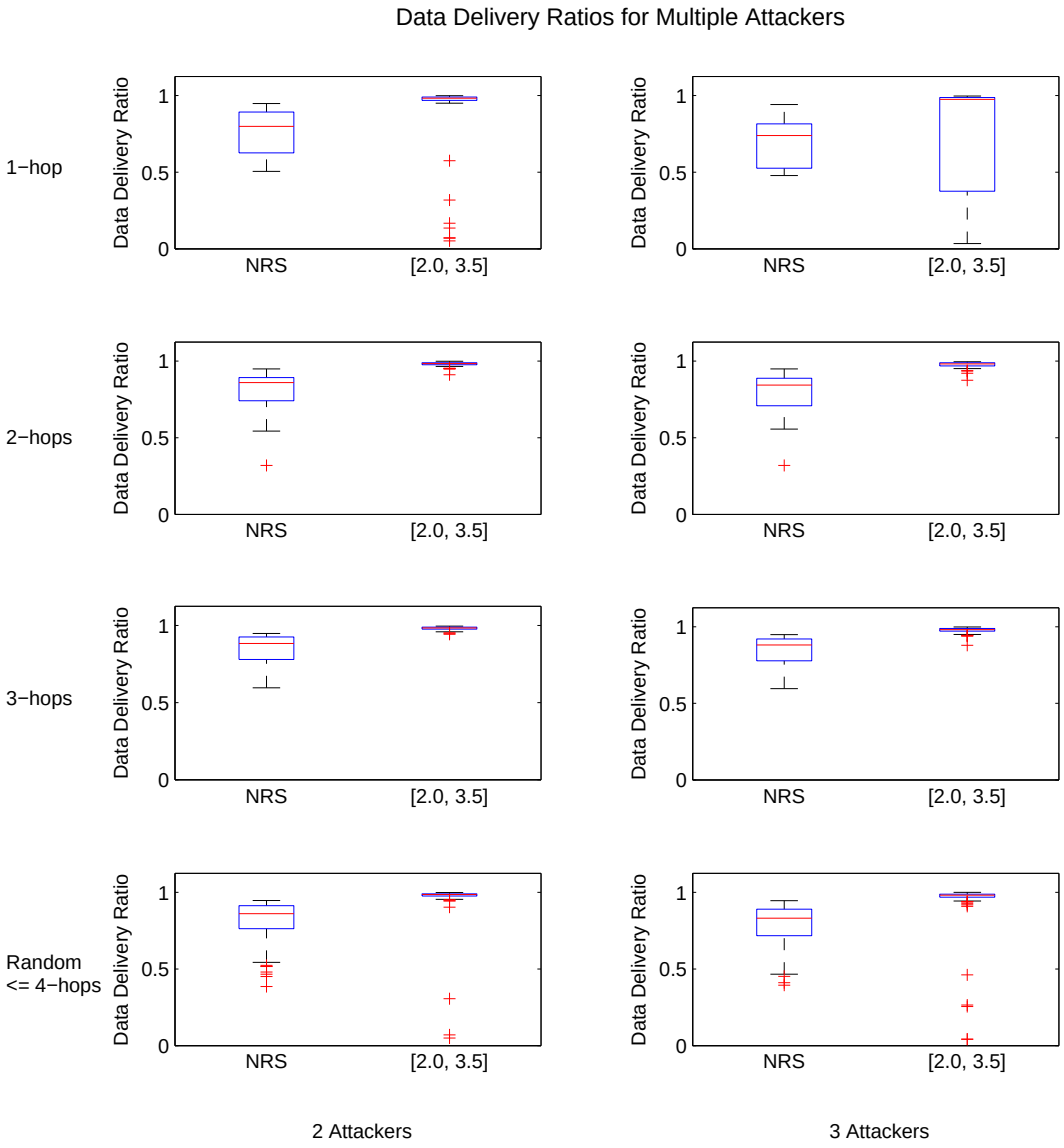


Figure 4.7: Data delivery ratios for multi-attacker scenarios

## 4.6 Alternative Route Opportunities via Additional Sinks

The purpose of the last simulation group was to determine whether CTP's inherent multi-gateway support could be utilized to yield even more opportunities for alternate routes around attacking/failing nodes. As previous simulations have been run with only a single gateway, a set of baseline simulations was first performed on the 100 node grid toplogy with multiple gateways (placed in the corners of the network), in order to gain better insight as to how additional gateways might provide additional attack resilience. The results of these simulations are shown below, in Figure 4.8.

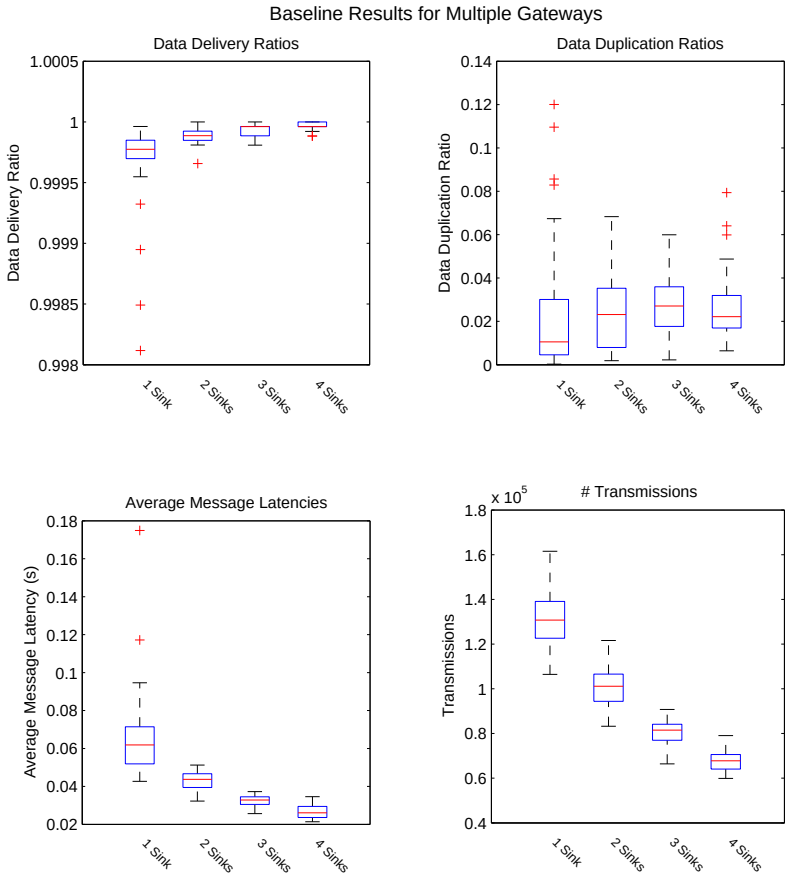


Figure 4.8: CTP performance with multiple gateways

As the additional gateways are added to the network, we see the slight improvements in the already near-ideal data delivery ratio. However, as the routing path lengths decrease with the introduction of gateways, we see the average message latencies

and number of transmissions decrease significantly. In these situations, the gateways effectively divide the network into "subnetworks;" this would significantly influence the impact of an attacker close to one of the gateways, as less data would be flowing to that gateway.

To test this hypothesis, two attacker placements are used:

- *Surrounding Attackers*: All nodes 3-hops inward from one of the gateways
- *Splitting Attackers*: All nodes in columns 3 and 4 in the grid topology

Both of these networks are associated with significant compromise; we expect poor data delivery ratios for only 1 sink, and increasingly better data delivery ratios as the additional sinks are added. The data delivery ratios for the *Surrounding* and *Splitting* cases are presented in Figures 4.9 and 4.10, respectively.

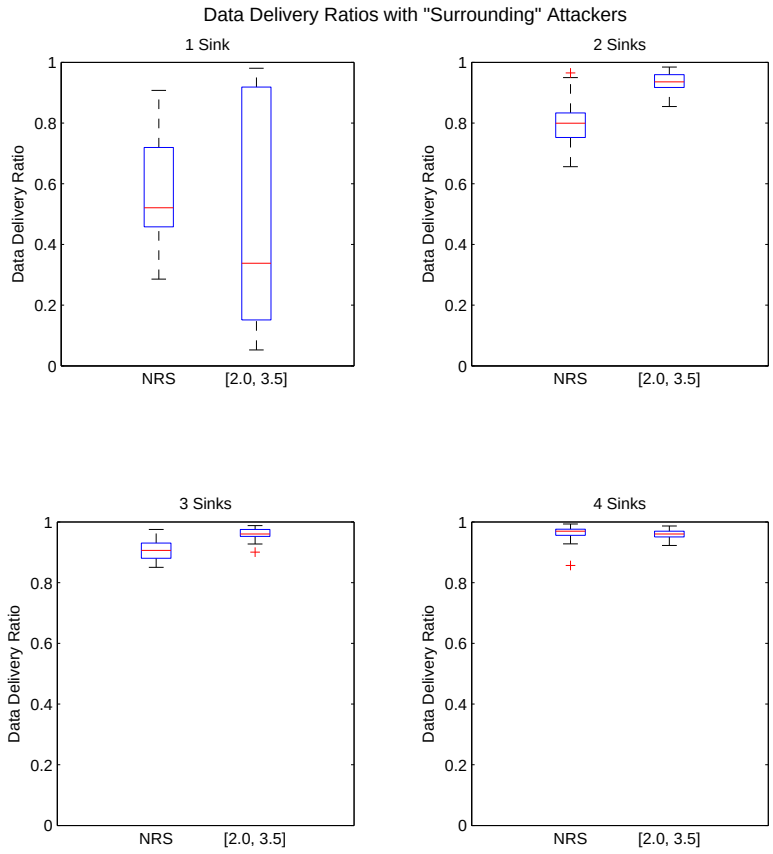


Figure 4.9: Data delivery ratios for multiple gateways and *Surrounding* attackers

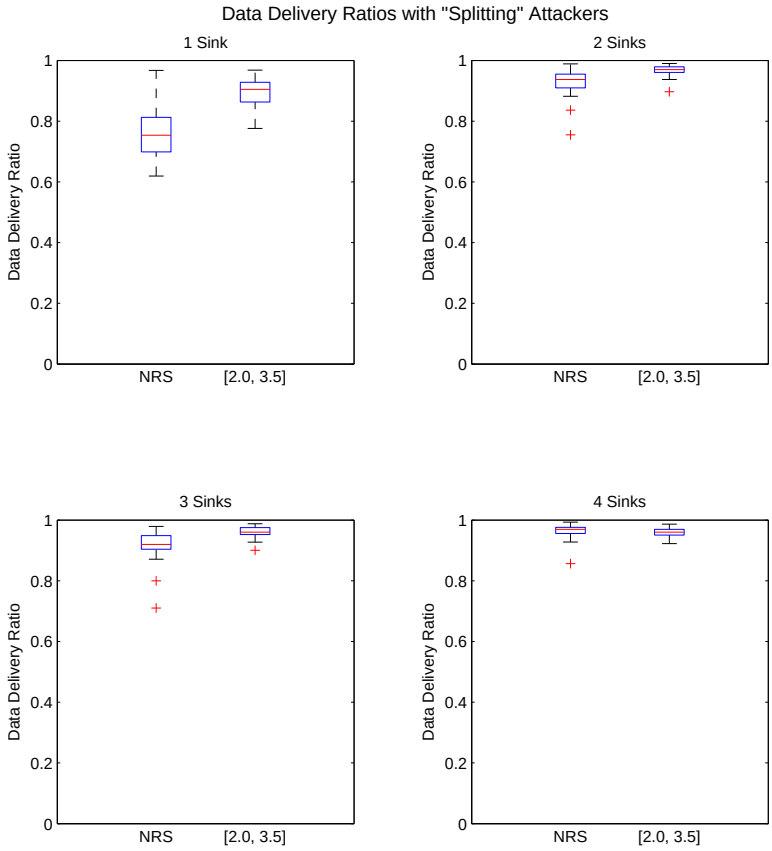


Figure 4.10: Data delivery ratios for multiple gateways and *Splitting* attackers

As expected, the single sink cases are associated with poor data deliveries. This is worse in the *Surrounding* cases, where blacklisting eliminates the only viable routes to the single gateway. Surprisingly, significant data recovery is still achieved in a number of of single sink cases; this may be attributed to the use of the sink node as a primary or secondary parent, despite unfavorable link qualities[9].

From these plots, we see that the addition of gateways alone does indeed reduce the impact of attackers. This is particularly true in the *Splitting* scenarios, as the attacking/failing nodes in columns 4 and 5 of the grid topology are likely to become leaf nodes. As the reactive schemes is introduced, we see additional data recovery achieved; it is clear that increasing routing opportunities through the use of multiple gateways complements our reactive scheme.

The overheads associated with these simulations are shown in the remaining plots in this section. These plots compare the baseline CTP performance (with no attacks) to the associated case including our reactive mechanism and attacks, with the intention of showing the cost of achieving the data recovery shown in Figures 4.9 and 4.10. These overheads follow similar trends to that observed in the previous simulation results; in cases where few to no alternate routes are available, the reactive scheme tends to produce excessive traffic, increasing overheads by an order of magnitude. As alternative routes are made available via the introduction of multiple gateways, we see data duplication, message latency and the number of transmissions decrease toward reasonable overheads.

As shown in Figures 4.11 and 4.12, average message latencies decrease as expected with the introduction of additional gateways. The outliers generally correspond to messages being routed during the transient period where a different gateway starts being utilized. The existence of slightly higher outliers in the 4 Sinks plots (as compared to the 3 Sinks plots) may be indicative of nodes switching back and forth between multiple gateways. As a simple hysteresis mechanism to prevent this, CTP requires a neighbor to have an ETX 2.0 better than the current parent to warrant a switch. However, it is likely that this condition will occur multiple times during a transient period when upstream nodes are resolving the "best" paths to sinks. This also suggests a limit to the benefits obtained by adding additional sinks to a network; adding too many gateways may result in leaf nodes switching paths fairly frequently until a stable state is finally reached.

[9]Recall the candidate score = 0 case discussed in Section 3.3.2.

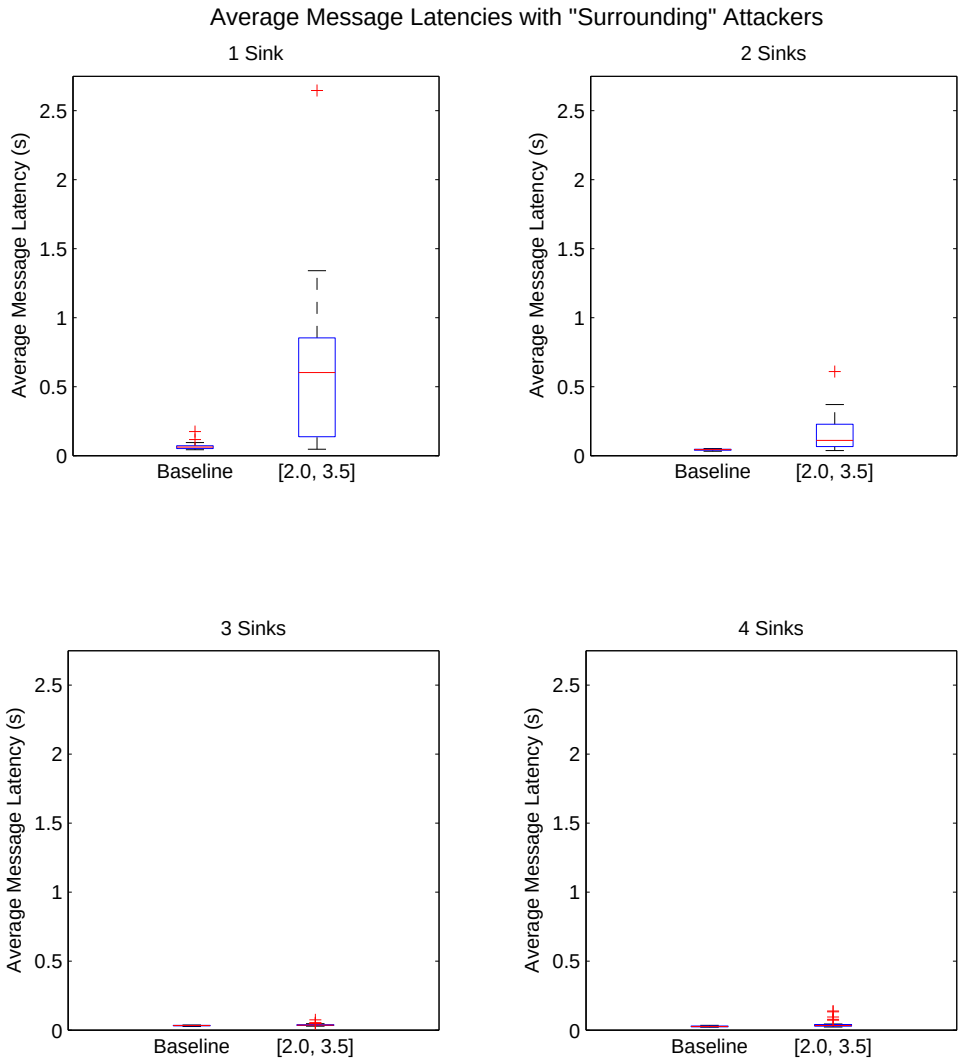


Figure 4.11: Average message latencies for multiple gateways and *Surrounding* attackers

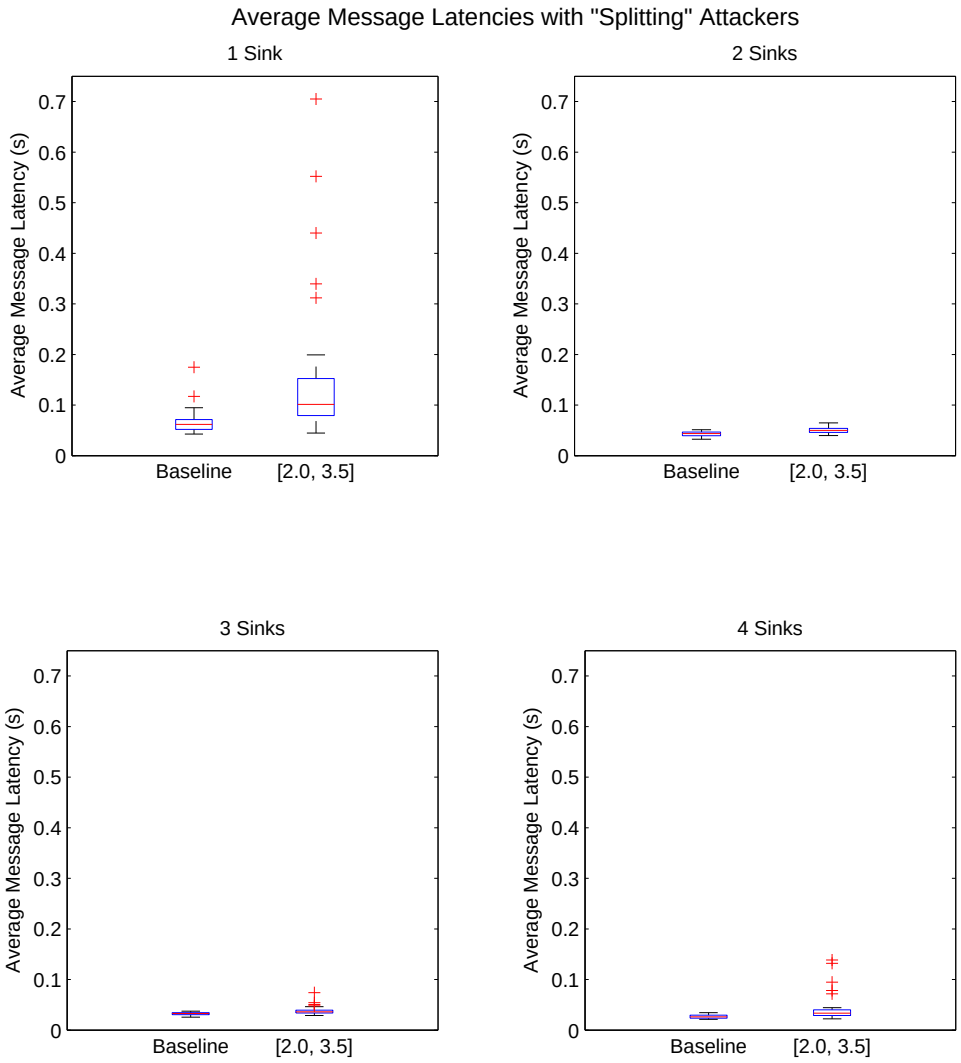


Figure 4.12: Average message latencies for multiple gateways and *Splitting* attackers

From the data duplication plots in Figures 4.13 and 4.14, we see that that the addition of an additional gateway alleviates the strain on the network imposed through the reactive process. However, adding the third and fourth gateways does not further minimize data duplication, indicating a lower limit. Based upon the medians in these plots, it appears that while responding to ongoing attacks, data duplication will increase from approximately 0.03 to 0.09. If every message in the network had been duplicated twice (perhaps via a traditional multipath scheme), this overhead would have been 1.00. Therefore, by reducing the number of nodes impacted by the attacker, the addition of gateways into the network alleviates the need to duplicate a large amount of data. However, additional gateways alone cannot reduce data duplication in the affected areas of the network; the reactive scheme will duplicate data as needed in these areas in an attempt to ensure data delivery.

The transmissions counts shown in Figures 4.15 and 4.16 generally appear to mirror the trends shown in the average message latency plots – significant improvement occurs with the introduction of the second gateway, and moderate improvement occurs with each added gateway.Furthermore, the trends present in 4.8 are also present. Because each additional gateway effectively reduces the number of nodes downstream from the attacker, fewer nodes are affected by the reaction process. With fewer nodes reacting, routing beacon and secondary parent traffic is lessened.

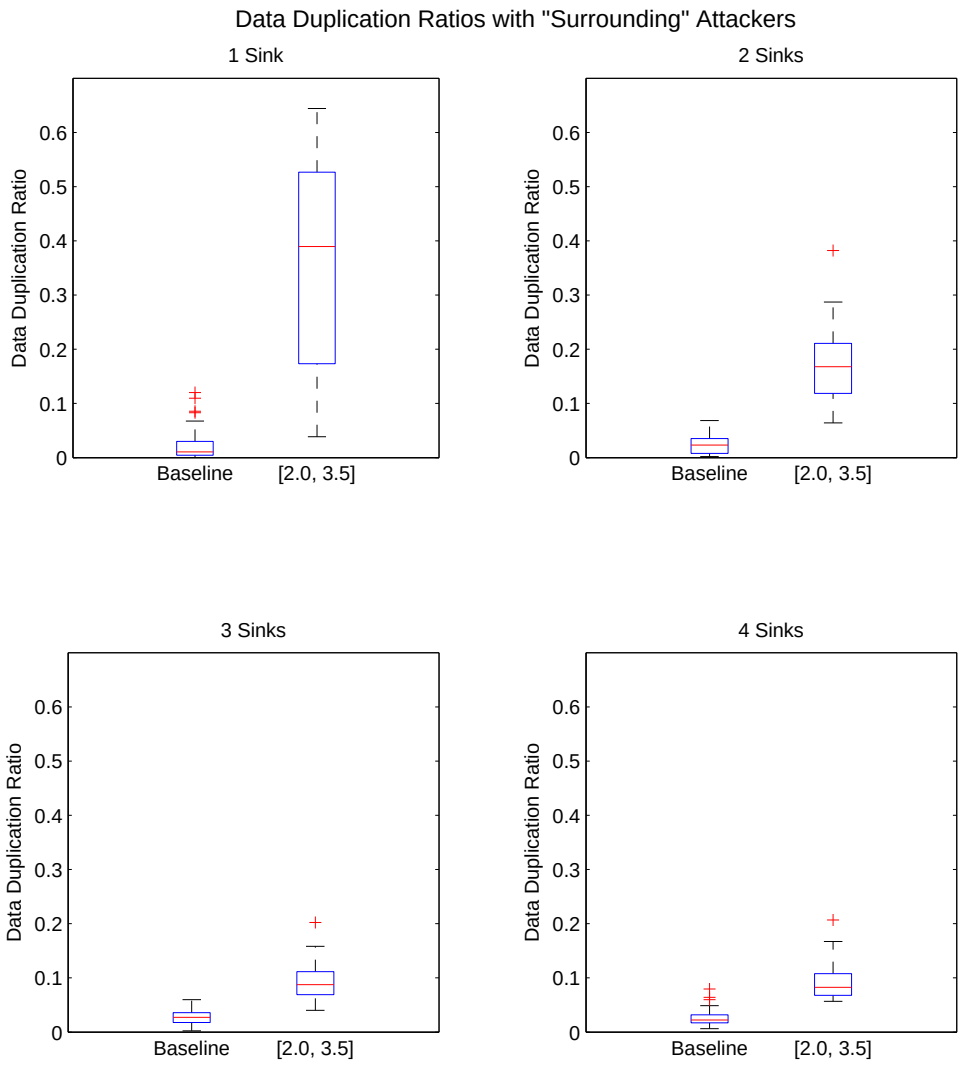


Figure 4.13: Data duplication ratios for multiple gateways and *Surrounding* attackers

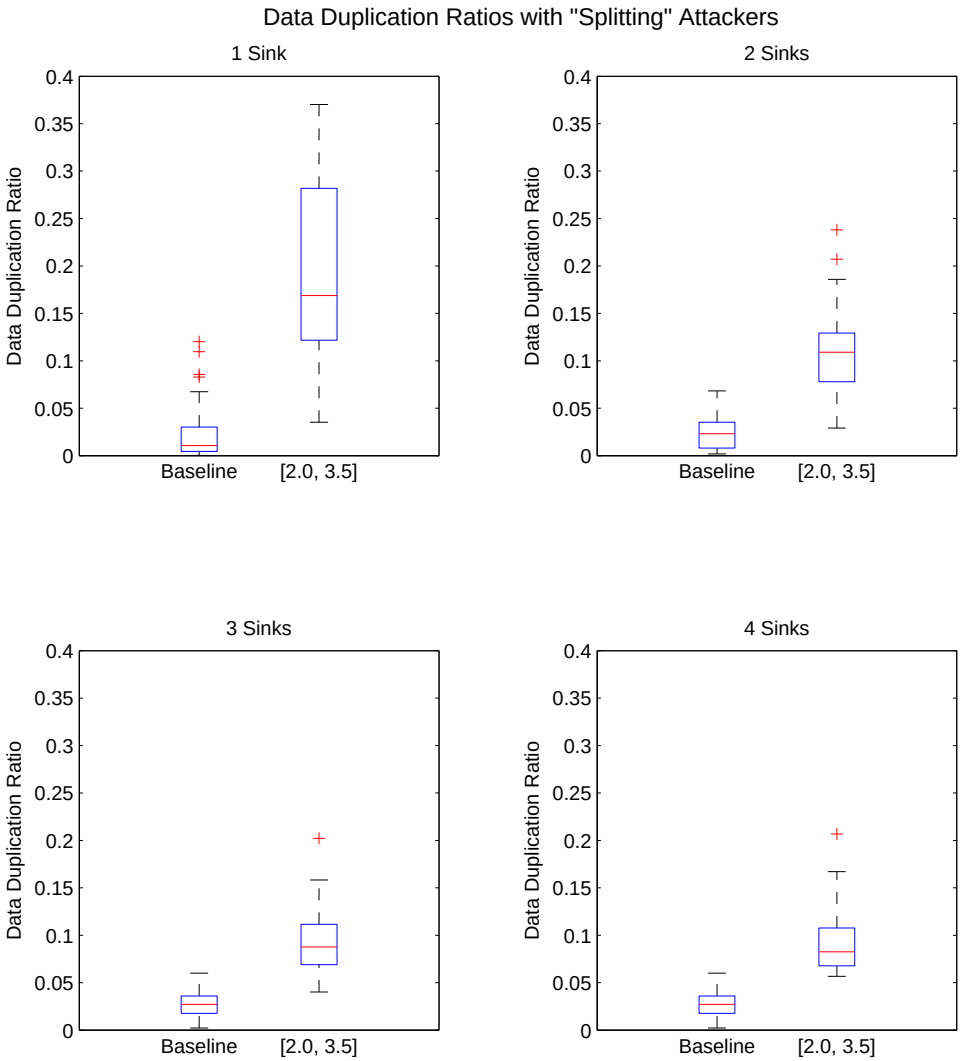


Figure 4.14: Data duplication ratios for multiple gateways and *Splitting* attackers

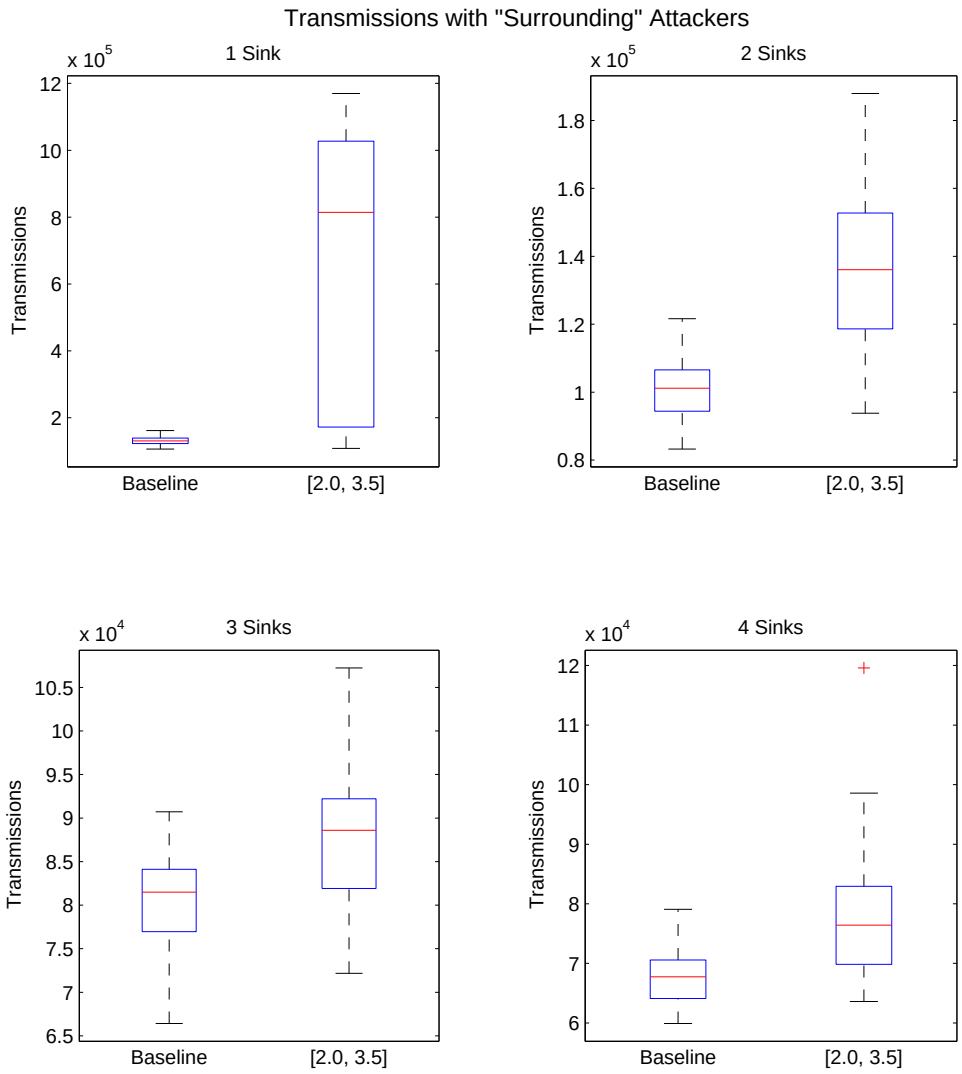


Figure 4.15: Transmission counts for multiple gateways and *Surrounding* attackers

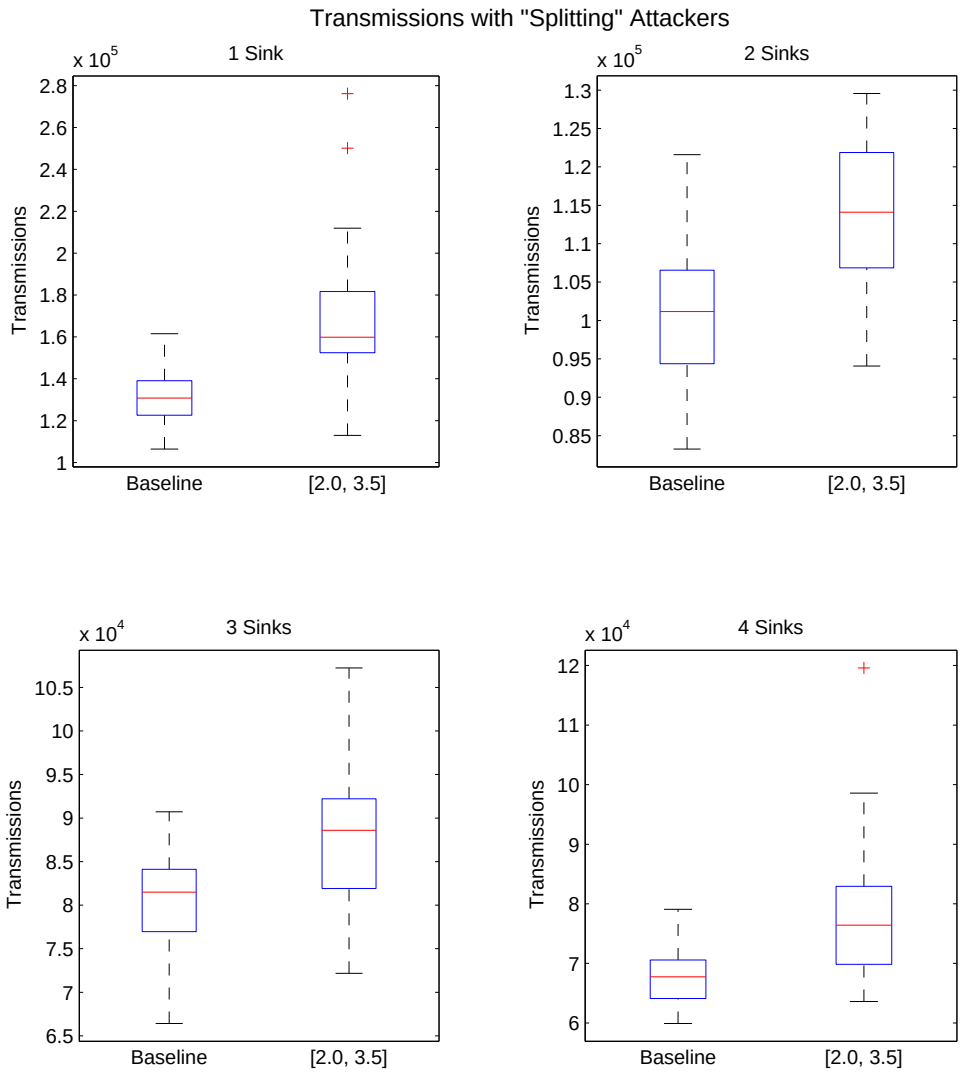


Figure 4.16: Transmission counts for multiple gateways and *Splitting* attackers

The results of the multi-gateway scenarios confirm the underlying themes that have motivated multipath routing, and clearly illustrate significant attack-resilience benefits. However, unlike traditional multipath schemes where data is always duplicated, our distributed reactive scheme provides resilience to ongoing attacks with significantly less data duplication (also implying fewer transmissions), because data is duplicated only as needed. As a trade-off for fewer transmissions, our scheme may result in some data loss prior to detection. However, by carefully tuning the $\alpha_{PRE}$ and $w_{PRE}$, a system designer could minimize this loss.

Although an accurate comparison of our scheme and traditional multipath schemes would require the implementation of additional routing protocol simulation schemes, simple estimates verify that our scheme yields fewer overheads. With our scheme's data duplication of 0.09 for 4 sinks, an average hop count of 2.5, and an average of 1.35 retransmission attempts per hop, and 80 messages, we estimate the number of transmissions as:

$$[1.00 + 0.09][(80)(1.35)(2.5)] = 294.3 \text{ transmissions} \tag{4.4}$$

However, if a multipath scheme were to duplicate every message along a second disjoint path, the value of our data duplication metric would be 1.00 (assuming delivery of every message). Using this value in the same calculation, we see a significantly larger transmission count:

$$[1.00 + 1.00][(80)(1.35)(2.5)] = 540.0 \text{ transmissions} \tag{4.5}$$

When projected over a larger period of time, where the time spent reacting to an attack is significantly less than the network uptime, we would expect the difference between these estimated transmission counts to grow linearly. Based upon these estimates, we conclude that this distributed reactive scheme yields significantly fewer overheads than a traditional multipath scheme, as data duplication is limited to the time required to react to an ongoing attack. Assuming this reaction time will be bounded, our scheme should always yield less data duplication and fewer transmissions.

# Chapter 5

# Conclusions and Future Work

One of the major contributions of this work has been the verification that a reactive, autonomous, and distributed routing scheme can yield significant data recovery against selective forwarding attacks, without requiring nodes to exchange auxiliary trust or detection information. Furthermore, we have shown how such a scheme may be integrated into the publicly available and well-used Collection Tree Protocol, making re-use of existing design aspects where possible. As indicated by our results, for properly selected thresholds, our scheme has potential to perform with only a modest introduction of overheads, because of the transitory nature of the secondary parent functionality. This scheme utilizes the multipath routing notation of achieving attack resilience through data duplication, but only when an attack is occurring, yielding fewer overheads under normal circumstances. Furthermore, we have provided some insight into how the use of multiple gateways in network can yield even greater resilience to selective forwarding attacks.

We believe that our scheme would greatly complement the SARP protocol presented in [22]. First, the introduction of our secondary parent functionality could allow for some additional data recovery while a node's trustworthiness is still being determined and compared against other nodes' reports. Secondly, our choice of CTP as the base protocol provides inherent improvements over [22]'s static hop counts and single gateway assumptions. Furthermore, if the trustworthiness of CTP data and routing frame fields could be monitored and established using a framework similar to that of SARP, we could begin addressing more realistic threat models in which attackers manipulate data and routing frame fields.

As noted in a number of various sections, some limitations of our current scheme provide a number of opportunities for future work and improvements. The results

presented in Chapter 4 were indicative of a need for a "negative feedback" protocol mechanism to ensure that frequent false positive assertions of the *Suspicious* or *Blacklist* states do not adversely affect network performance. In cases where such false positives hindered data delivery, we also saw increased message latencies and data duplication ratios. Because these symptoms are observable from gateways, it may be possible to generate alerts to network administrators, or perform automated recovery. One possible method for automated recovery is the use of an authenticated "backoff" broadcast from a gateway that informs nodes to cease the usage of secondary parents, un-blacklist nodes, or to modify their $T_{Suspicious}$ and $T_{Unreliable}$ thresholds. In order to ensure that such a feature is not abused, the usage of an authentication scheme presented in [12] or [33] is highly recommended.

A simpler mechanism to recover from false positive blacklisting may include the use of periodic clearing of a blacklist. Depending upon the criticality of the network and the value of the data, this period may be short or long. A short period would allow attackers re-entry into victims' routing tables, but also reduce the effect of false positives. On the other hand, a longer window would decrease the amount of data an attacker could drop over time, and the risk of preventing valid nodes from being allowed back into routing tables.

Another possible modification might include nodes adapting their thresholds as link qualities change over time, or to observe local changes in message delivery as they begin using secondary parents or blacklist parent nodes. CTP routing frames contain a "C-bit" that denotes when a node is congested. The 2007 draft of The Network Working Group's TEP123 [32] states that the associated congestion condition occurs when a CTP node drops a data frame[1]. If nodes were to utilize a congestion status mechanism in the routing protocol, they could potentially start with high thresholds and decrease them over time until congestion occurs. Once congestion starts occurring, nodes could then revert to the previous thresholds and increase their transmission timer (perhaps in an exponential backoff fashion).

As briefly mentioned in Section 3.2.2, the PRE Cache may also provide opportunities for the detection of a data tampering attack. If (at the cost of additional memory overhead) data payloads or message authentication codes were to be stored in the cache, a match on the data origin and sequence number, but conflict on the data (or MAC), would indicate data tampering. Future investigations might consider whether

[1]However, at the time of writing the TinyOS CTP implementation does not assert the C-bit.

immediate blacklisting is appropriate, or determine an appropriate systematic approach to manipulating a node's PRV value for this case. Additionally, an improved message-matching scheme could be investigated to account for cases where an attacker may manipulate the data origin and sequence number fields.

To better determine how our work may be integrated into other protocols, such as SARP, an understanding of performance and limitations with respect to more advanced threat models may prove worthwhile. Such attack models include nodes that vary their attack rates and reduce attack rates upon observing children choose secondary parents, nodes that cease routing beacons to become removed from routing tables in order to rejoin and have their PRV re-initialized, and nodes that tamper with CTP fields.

To achieve better performance against attackers dropping data at a lower rate, the reactive scheme presented in this work could be extended to maintain PRVs for various priorities/classes of messages. Based upon the importance and frequency of each class of data, different $\alpha_{PRE}$ values could be used to allow nodes to react more quickly to parents dropping particular classes of data. Consider a network in which nodes periodically generate low-priority data, as a normal network status "heart-beat," and generate high-priority data messages when important events occur in the monitored environment. A node could then maintain three PRV values: one for low priority messages, one for high priority messages, and another for all priority classes. If an attacker only drops high-priority messages, the low-priority and all-priorities PRVs may not be affected, but the high-priority PRV would provide an indication of a *Suspicious* or *Unreliable* parent. A simple $max()$ function of all three PRVs could be used to determine which PRV to compare to thresholds.

# Bibliography

[1] Kirk Martinez, Jane K. Hart, and Ong Royan. Sensor network applications. *Computer*, 37(8):50–56, 2004.

[2] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, August 2008.

[3] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, March 2002.

[4] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *Computer Communication Review*, 39(1):50–55, 2009.

[5] Mohammad Mehedi Hassan, Biao Song, and Eui-Nam Huh. A framework of sensor-cloud integration opportunities and challenges. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, pages 618–626. ACM, 2009.

[6] Werner Kurschl and Wolfgang Beer. Combining cloud computing and wireless sensor networks. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services (iiWAS '09)*, pages 512–518, New York, New York, USA, 2009. ACM Press.

[7] Tanya Roosta, Shiuhpyng Shieh, and Shankar Sastry. Taxonomy of security attacks in sensor networks and countermeasures. In *Proceedings of the First IEEE Conference on System Integration and Reliability Improvements*, pages 13–15, 2006.

[8] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2-3):293–315, September 2003.

[9] Yih-Chun Hu, Adrian Perrig, and Daniel B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, February 2006.

[10] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2003 ACM workshop on Wireless security (WiSe '03)*, pages 30–40, San Diego, CA, USA, 2003.

[11] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):500–528, 2006.

[12] Peng Ning, An Liu, and Wenliang Du. Mitigating DoS attacks against broadcast authentication in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(1):1–35, January 2008.

[13] An Liu and Peng Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track*, pages 245–256. IEEE, April 2008.

[14] Ju-hyung Son, Haiyun Luo, and Seung-Woo Seo. Denial of service attack-resistant flooding authentication in wireless sensor networks. *Computer Communications*, 33(13):1531–1542, 2010.

[15] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the Third International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pages 259–268, Berkeley, California, USA, 2004.

[16] Xin Hu, Taejoon Park, and Kang G. Shin. Attack-tolerant time-synchronization in wireless sensor networks. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM)*, pages 41–45. IEEE, April 2008.

[17] Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. Anomaly detection in wireless sensor networks. *IEEE Wireless Communications*, 15(4):34–40, 2008.

[18] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the IEEE Symposium on*

*Security and Privacy (S&P'05)*, pages 49–63, Oakland, California, USA, 2005. IEEE Computing Society.

[19] Ana Paula R. da Silva, Marcelo H. T. Martins, Bruno P. S. Rocha, Antonio A. F. Loureiro, Linnyer B. Ruiz, and Hao Chi Wong. Decentralized intrusion detection in wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks (Q2SWinet '05)*, pages 16–23, New York, New York, USA, 2005. ACM Press.

[20] Chin-Yang Tseng, Poornima Balasubramanyam, Calvin Ko, Rattapon Limprasittiporn, Jeff Rowe, and Karl Levitt. A specification-based intrusion detection system for AODV. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks (SASN '03)*, pages 125–134, New York, New York, USA, 2003. ACM Press.

[21] Eliana Stavrou and Andreas Pitsillides. A survey on secure multipath routing protocols in wsns. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 54(13):2215–2238, September 2010.

[22] Yan Sun, Kenneth Jr. Rahn, Rajini Anachi, Onur Savas, and Lisa Cingiser DiPippo. Secure adaptive routing protocol for wireless sensor networks. Technical report, University of Rhode Island, Department of Computer Science, 2010.

[23] Bai Li, Robin Doss, Lynn Batten, and Wolfgang Schott. Fast recovery from node compromise in wireless sensor networks. In *Proceedings of the 3rd International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–6. IEEE, 2009.

[24] Adrian Perrig, Robert Szewczyk, J.D. Tygar, Victor Wen, and David E. Culler. SPINS: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[25] Jing Deng, Richard Han, and Shivakant Mishra. INSENS: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2):216–230, January 2006.

[26] B Xiao, B Yu, and C Gao. CHEMAS: Identify suspect nodes in selective forwarding attacks. *Journal of Parallel and Distributed Computing*, 67(11):1218–1230, November 2007.

[27] Young Ki Kim, Hwaseong Lee, Kwantae Cho, and Dong Hoon Lee. CADE: cumulative acknowledgement based detection of selective forwarding attacks in

wireless sensor networks. In *Proceedings of the Third International Conference on Convergence and Hybrid Information Technology*, pages 416–422, 2008.

[28] Nidal Nasser and Yunfeng Chen. SEEM: Secure and energy-efficient multipath routing protocol for wireless sensor networks. *Computer Communications*, 30(11-12):2401–2412, September 2007.

[29] Wenjing Lou and Younggoo Kwon. H-SPREAD: A hybrid multipath scheme for secure and reliable data collection in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 55(4):1320–1330, July 2006.

[30] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pages 1–15, New York, New York, USA, 2009. ACM Press.

[31] Ugo Colesanti and Silvia Santini. A performance evaluation of the collection tree protocol based on its implementation for the castalia wireless sensor networks simulator. Technical report, ETH Zurich, Department of Computer Science, 2010.

[32] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Sukun Kim, Philip Levis, and Alec Woo. TEP 123: The collection tree protocol (CTP). Technical report, TinyOS, 2007.

[33] Sushil Jajodia, Sencun Zhu, Peng Ning, and Donggang Liu. Practical broadcast authentication in sensor networks. In *Proceedings of The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '05)*, pages 118–132, 2005.

# Appendix A

# Modified CTP Snoop Interface Usage

Pseudo code for modifications to CTP's usage of the TinyOS Snoop Interface is presented below. These modifications allow a node to determine if it has been selected as a secondary parent, as well as monitor the parent node's retransmissions.

```
event Snoop.receive(message)
{
    // Secondary parent reception
    // (Modification to CTP)
    if(message.secondary_parent == my_addr)
        // Process message as if it were unicast to us
        receive(message);
    endif

    // Monitor parent retransmissions
    // (Modification to CTP)
    if(message.source == my_parent && pre_cache.contains(message))
        // Optionally, can add a message validity check here
        mark_forwarded(pre_cache, message);
    endif

    // Trigger route update if a neighbor broadcast with pull-bit (P) set
    // (Standard CTP behavior)
    if(message.bits & PULL_BIT)
        perform_route_update();
    endif
}
```

# Appendix B

# Modified Route Update Procedure

The below pseudo code describes the modified CTP route update procedure designed to avoid inducing routing loops. For brevity, functionality pertaining to routing table maintenance, congestion, and adaptive beaconing is not shown here. The functionality of *compute_candidate_score()* is represented by Table 3.2. Note that this pseudo-code is intended to illustrate the logical flow of the route update procedure; it is not optimized for complexity and minimum memory utilization.

```
task route_update(parent_status)
{
    list        candidates;
    route_entry best;

    loop: entry in routing_table
        if(entry.parent != INVALID_ADDR && entry.parent != my_addr)
            entry.link_etx = LinkEstimator.get_link_quality(entry);
            entry.path_etx = entry.etx + entry.link_etx;

            if(entry.link_etx > BAD_LINK_THRESHOLD)
                continue;
            else
                if(parent_state == SUSPICIOUS || parent_state == UNRELIABLE)
                    entry.score = compute_candidate_score(entry);
                endif

                candidates.add(entry);

            endif
        endif
```

```
        // Standard CTP behavior
        if(parent_status == NORMAL)
            best = sort_by_etx(candidates);

            // Avoid bad entries and only switch if
            // neighbor's ETX is significantly better
            if(best != INVALID_ADDR &&
               parent.path_etx - best.path_etx > SWITCH_THRESH)
                parent = best
            endif

            secondary_parent = NO_SECONDARY_NEEDED;

        // Parent is suspicious or unreliable.
        else
            // Sorting precedence: first score, then ETX
            best = sort_by_score_and_etx(candidates);

            if(best != DO_NOT_CONSIDER)
                // Advertise lack of (secondary) parent to neighbors
                if(parent_status == UNRELIABLE)
                    parent = INVALID_ADDR;
                else
                    secondary_parent = NO_SECONDARY_FOUND;
                endif

            else
                if(parent_status == UNRELIABLE)
                    parent = best;
                else
                    if(parent.path_etx - best.path_etx > SWITCH_THRESH)
                        parent = best;
                    else
                        secondary_parent = best;
                    endif
                endif
            endif
        endif
    endloop
}
```